University of Alberta

Trajectory Optimization for Flat Dynamic Systems

by

Sachin Kansal

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**.

in

Process Control

Department of Chemical and Materials Engineering

Edmonton, Alberta
Fall 2000

National Library of Canada

Acquisitions and Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada

Acquisitions et services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-59823-3

Canada

University of Alberta

Library Release Form

Name of Author: Sachin Kansal

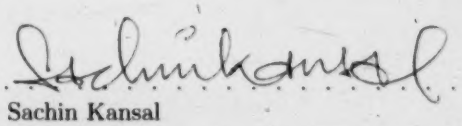
Title of Thesis: Trajectory Optimization for Flat Dynamic Systems

Degree: Master of Science

Year this Degree Granted: 2000

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided, neither the thesis nor any substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.



Sachin Kansal
Imperial Oil
Products and Chemicals Division
453 Christina Street South
PO Box 3022
Sarnia, ON N7T 8C8
CANADA

Date: May 10, 2000

Sometimes a scream is better than a thesis!
- Ralph Waldo Emerson (1803-1882)

Abstract

A substantial class of chemical manufacturing processes are operated in a transient manner and cannot be considered to reach a steady-state. The optimization of such processes requires determination of optimal time-varying trajectories for the manipulated variables. A dynamic optimization problem thus needs to be solved that, in majority of cases, involves differential-algebraic constraints on the manipulated and state variables. In general, such optimal control problems are difficult to solve and solution schemes are based on approximate and computationally demanding discretization methods.

This thesis proposes a new method to solve a class of dynamic optimization problems in which the nonlinear differential-algebraic process model is *flat*. The method exploits, as appropriate, the *differential flatness* or *orbital flatness* of dynamics of the process model to explicitly eliminate the differential state equations from the dynamic optimization problem. This enables the representation of the optimization problem in a set of new coordinates of *flat outputs*, in which the problem is purely algebraically constrained. This transformed optimization problem is then shown to be readily solvable using one of many available optimization codes.

The proposed Normalized Dynamic Optimization (NDO) approach is shown to obtain accurate and efficient solutions to a range of benchmark problems taken from dynamic optimization literature. In view of the encouraging results obtained from the application of the algorithm to off-line dynamic optimization problems, the thesis also makes a preliminary attempt to investigate the applicability of the proposed algorithm to real-time optimization.

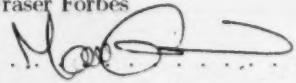
Finally, the algorithm is summarized and evaluated in terms of its efficiency to solve the class of benchmark optimization problems. Noting that the application area of this algorithm is presently restricted to *flat* process models, further research directions are also identified that may broaden the applicability of the general concept.

University of Alberta

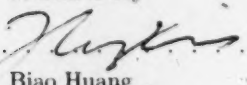
Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Trajectory Optimization for Flat Dynamic Systems** submitted by Sachin Kansal in partial fulfillment of the requirements for the degree of **Master of Science** in *Process Control.*

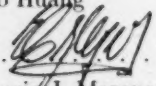
J. Fraser Forbes



Martin Guay



Biao Huang



Horacio J. Marquez

Date: May 10, 2000

To

Mom and Dad

and

Bhaiya and Bhabhi

Acknowledgements

I would like to express my gratitude to Drs. J. Fraser Forbes and Martin Guay for their guidance and support throughout the course of this work. I am indebted to them for their in depth discussions, comments and useful insights. I have to thank Dr. Martin Guay for his assistance in the more troublesome of the mathematical complications that I have come across, and to Dr. Forbes to keep me focused. I would also like to thank Dr. Forbes for making my transition into life in Canada much smoother, as well as his constant help and support during my university and career years.

The financial support of the *Natural Sciences and Engineering Research Council of Canada* is gratefully acknowledged, as is the enthusiasm and constant encouragement of the faculty, staff and graduate students at the *Department of Chemical and Materials Engineering*. A special thanks goes to my sis Abha in India and Shirls, who were always there for me when I needed them. Last but not the least, I would like to thank all my friends at the *U of A*: Rajesh, Rohit, Nikhil, Ashish, Bhushan, Srini, Misha, Ashutosh, Al, Andrew, Ryan, Steve, Safia, Shaista, Sonia, Puja, Nehal and so many more, for the good times we had at *The White House*, *The Powerplant* and on *Whyte* among other places.

Contents

List of Figures 1

List of Tables 2

1 Introduction 1

1.1 Batch Process Characteristics 2
1.2 Optimization Problem Formulation 3
1.3 Solution Techniques 5
1.4 Comparative Discussion 6
1.5 Thesis Contributions 8
1.6 Thesis Organization 9

2 Nonlinear Dynamic Systems 11

2.1 Dynamic Control Systems 12
2.2 Differential Geometric Approaches 13
2.2.1 State Feedback Linearization 15
2.2.2 Input-Output Linearization 16
2.2.3 Approximate Linearization 16
2.3 Differential Flatness 17
2.3.1 Illustrative Example 18
2.4 Orbital Flatness 19
2.4.1 Time Scaling Transformations 20
2.4.2 Illustrative Example 21
2.5 Characterization of Flatness 23
2.5.1 Reduction of System Codimension 24
2.5.2 The Gardner and Shadwick (GS) Algorithm 25
2.5.3 Linearization by Endogenous Dynamic Feedback 25
2.5.4 An Algorithm for Orbital Feedback Linearization 26
2.6 From Flatness to Trajectory Generation 26

3 Trajectory Optimization 28
3.1 Trajectory Optimization Problem 29
3.2 Normalized Form Optimization 30
3.3 Steps in the NDO Algorithm 32
3.3.1 Flatness and System Transformation 33
3.3.2 Normalization of the DAOP Problem 37
3.3.3 Parameterization and Trajectory Determination 38
3.4 Illustrative Examples 39
3.4.1 Single Integrator System 39
3.4.2 Parallel Reaction Problem 41
3.4.3 Consecutive Reaction Problem 43
3.4.4 Crane Container Problem 45
3.5 Summary 49
3.6 Conclusions 49

4 Real-Time Trajectory Optimization 51
4.1 Introduction 51
4.2 Real-Time Optimization of Batch Processes 54
4.2.1 MPC and Dynamic Real-Time Optimization 55
4.2.2 DRTO - Problem Formulation as EOTs 56
4.3 EOTs in the NDO Framework 58
4.4 Illustrative Examples 60
4.4.1 Problem Scope and Solution Assumptions 60
4.4.2 Single Integrator System 60
4.4.3 Crane Container Problem 63
4.4.4 Continuous Stirred Tank Reactor 66
4.5 Summary of Results 71

5 Summary and Research Directions 73
5.1 Summary of Results 73
5.2 Future Research Directions 74
5.2.1 NDO Algorithm Issues 74
5.2.2 Real-Time Implementation Issues 75

Nomenclature 77

Bibliography 80

A Language of Exterior Calculus 86
A.1 Exterior Algebra 86
A.2 Differentiable Manifolds 87

A.3 Differential Forms 88
A.4 Exterior Derivative and Differential Systems 88
A.5 Pfaffian and Control Systems 88
A.6 Integrability and Congruence of Forms 89
A.7 Derived Systems and Differential Flatness 90

B Algorithms for Characterization of Differential Flatness **91**
B.1 Reduction of System Codimension 91
B.2 Gardner and Shadwick (GS) Algorithm 92
B.3 Linearization by Endogenous Dynamic Feedback 93

List of Tables

1.1 Comparison of Algorithms for Solution to DAOPs. 8

3.1 Comparison of objective function values for illustrative examples. 50

4.1 Results of Real-Time Optimization on the Single Integrator System. 62
4.2 Results of Real-Time Optimization on the Crane Container Problem. 66
4.3 Model parameters of the CSTR [Rothfuss et al., 1995 68
4.4 Results of Real-Time Optimization on the CSTR System. 71

List of Figures

2.1 Depiction of Feedback of Control Systems. 15

3.1 Comparison of Trajectories for the Single Integrator System. 41
3.2 Optimal Trajectories for the Parallel Reaction Problem. 43
3.3 Optimal Trajectories for the Consecutive Reaction Problem. 45
3.4 Constrained Trajectories for the Crane Container Problem. 49

4.1 Plant Decision-Making Hierarchy. 52
4.2 Typical model-based RTO system [Forbes, 1994]. 52
4.3 Depiction of the General Approach to MPC. 55
4.4 Structure of a Dynamic Real-Time Optimizer [Loeblein, 1997]. 56
4.5 Comparison of Input Trajectories for the Single Integrator System. 63
4.6 Comparison of Input Trajectories for the Crane Container Problem. 67
4.7 Comparison of Input Trajectories for the CSTR System. 71
4.8 Comparison of State Trajectories for the CSTR System. 72

Chapter 1

Introduction

Mothers are the necessity of invention. — Bill Watterson, "Calvin and Hobbes"

There is a constant drive towards improving the economic performance of process plants. Many of the recent developments in the chemical engineering literature have focused on determining the optimal steady-state operation of continuous processes. However, there are a number of important chemical engineering processes that are operated in a transient manner and cannot be considered to reach a steady-state. Of particular interest, among this class, is that of batch processes. Batch processing, because of its enormous flexibility and diversity, is extensively used for production in a variety of industries including specialty chemicals, pharmaceuticals, steel, polymers and so forth. Even in continuous operations, there are situations where the process is intentionally operated in a transient manner (*e.g.*, grade transitions). In such transient processes, steady-state optimization approaches cannot be applied and operations optimization needs to be considered within a dynamic framework. In addition, many of these transient processes are characterized by highly nonlinear models that increase the mathematical complexity of any optimization problem. These factors have contributed to the situation that the operation of these processes is, in general, seldom optimized. The processes are then carried out to a defined recipe which may have been arrived at during development and is never changed. This type of operation however, in most cases, is sub-optimal considering that the system is always subject to some disturbances.

Recognizing the need for optimization of transient processes, this thesis focuses on finding an accurate and efficient solution to the dynamic optimization problems posed by batch processes. A key characteristic of dynamic optimization problems is that the process model consists of differential equations. Numerical solution techniques for such problems (*e.g.*, dynamic programming, collocation on finite elements and so forth), which are currently in widespread use, are usually based on discretization schemes and can be computationally prohibitive. Furthermore, the techniques may require accurate initial estimates of the optimal process operation, before a satisfactory answer is obtained. This thesis proposes an alternative method to solve the dynamic optimization problems for processes whose

nonlinear process model is *flat*. The approach exploits, as appropriate, either the *differential flatness* or *orbital flatness* of the process model, to explicitly eliminate the differential equations from the optimization problem. The resulting optimization problem is a simpler, algebraically constrained problem that can be solved using readily available optimization codes.

The following sections describe the difficulties associated with the operation of batch processes[1], as well as define the associated dynamic optimization problem. An overview of the widely used techniques to solve such dynamic optimization problems is also presented before proposing the new technique.

1.1 Batch Process Characteristics

Batch processes are operated in a diverse array of situations (*i.e.*, production scales may range from thousands of tonnes to a few kilograms per year, products and raw materials may be extremely valuable or extremely cheap, limited availability and costs of energy or other resources such as manpower may be dominating or negligible, *etc.*). The processes may be *ill-defined*, which refers to the variability, uncertainty and lack of knowledge very commonly encountered in the design and operation of batch reactors. To compound the already difficult problem, processing facilities have to cope with many different products. In multi-product batch processing facilities, operation requires control strategies that can operate over a wide range of conditions. This diversity implies that separate optimal operating schemes have to be determined for these processes under different situations. The control of batch operations, thus, can have widely differing immediate objectives. These may consist simply of damping out the effects of small perturbations from known standard conditions, as in conventional control of continuous processes. In other cases, however, it would be required to develop a complete operating strategy on-line in response to observed system behaviour. Increasing competition in the process industries, however, presses for *optimal operation* in all situations. In other words, the problem of determination of optimal plant operations (or setpoint trajectories) for these so-called dynamic processes, gains a new importance.

Recognizing the need and importance, research in the area of batch process optimization has been rich and has centered around the following two lines of investigation: 1) The *optimal servo control* problem, which refers to the computation of optimal input trajectories that lead the system from certain initial conditions to final specified conditions; and 2) The *optimal regulatory control* problem that deals with selection of control strategies which reduce the effects of undesirable disturbances on process conditions. The servo problem has been studied under the following two subdivisions: 1) *Chemical control*, where the optimal policies are developed in order to keep some variables, such as compositions or reaction

[1] The techniques discussed in this thesis apply to batch processes, as well as a class of transient processes that may include start-up/shut down control schemes for continuous processes. However, all these processes will be collectively referred to as batch processes in this thesis.

rates, constant throughout the batch; and 2) *Optimal control*, which refers to the design of optimal trajectories that optimize certain objective functions. The regulatory problem, on the other hand, has been studied under the framework of *on-line* or *real-time optimization* and/or differential game theory. The common feature of all these optimal control methods is that they incorporate the knowledge of the dynamic and nonlinear process model to determine the optimal operating conditions for the process. The detailed reaction models for batch reactor problems are generally constituted by a set of highly coupled nonlinear ordinary differential and/or algebraic equations. The solution to the optimization problem with this dynamic model thus requires a considerable computational effort. The solution is further complicated by the effect of batch-to-batch variations (*e.g.*, due to raw material impurities or up-stream disturbances and so forth), that needs to be properly represented in the solutions [Terwiesch *et al.*, 1994].

Real-Time Optimization (RTO) is a concept that combines the above strategies. In RTO, the optimization of a detailed mathematical model of the process is performed on-line. The optimization function is usually economic, and the optimization problem consists of determination of set-points for the process controllers such that a objective function is optimized. The optimization is carried out in real-time, implying that information about the process is acquired during its operation in order to determine the current state of the process. The process model is optimized taking the collected on-line data of the process into account and the optimization result is implemented into the process changing the current operating set-point. The core of any RTO system is an algorithm that is used to solve the optimization problem several times during the process operation. For continuous steady-state optimization problems, the model complexity is not as critical, and hence a variety of algorithms are able to achieve this. For differential/algebraic nonlinear models, however, this problem becomes a lot more difficult and sometimes impossible to solve. This thesis takes a step in this direction by proposing an algorithm to solve a specific class of these problems efficiently. This algorithm is shown to give an accurate and efficient solution to several benchmark dynamic optimization problems from literature. The characteristics of the algorithm seem to make it particularly suited for on-line applications that require fast and accurate solution to the updated optimization problem at every optimization interval.

1.2 Optimization Problem Formulation

In dynamic optimization problems, an objective function is optimized with respect to dynamic model equations and other constraints. The class of processes considered in this thesis can be generically modeled with the following dynamic equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \qquad t \in [t_0, t_f] \tag{1.1}$$

where $\mathbf{x}(t) \in M \subset \Re^n$ is the vector of state variables and $\mathbf{u}(t) \in \mathcal{U}^p$ the vector of control variables that has to be chosen optimally over an admissible set of controls. The boundary

conditions generally depend on the physical nature of the problem. For example, specification of the composition of initial charge in a batch reactor would take the form:

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{1.2}$$

If the initial condition, such as the above, was to be fixed only, the result would be a straightforward initial value problem. In other practical systems, the final state may be constrained:

$$\mathbf{x}(t_f) = \mathbf{x}_f \tag{1.3}$$

(*e.g.*, the requirement that the final product be of a given composition); then the result is a two-point boundary value problem. The final state constraints represented above, along with other constraints such as the transversality conditions: $\psi(\mathbf{x}(t_f)) = 0$ *etc.*, can be studied under the following general representation of constraints:

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 \tag{1.4}$$

$$\mathbf{c}(\mathbf{x}(t), \mathbf{u}(t)) = 0. \tag{1.5}$$

In addition, constraints in the form of upper and lower bounds on the variables are common:

$$\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U \tag{1.6}$$

$$\mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U \tag{1.7}$$

where the subscripts, L and U refer to lower and upper bounds respectively[2]. In order to specify what is meant by optimal, an objective function:

$$\Phi(\mathbf{x}(t), \mathbf{u}(t)) = G(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} F(\mathbf{x}(t), \mathbf{u}(t))dt \tag{1.8}$$

is defined that has to be maximized or minimized. This formulation of the objective function is sufficiently general that it allows the treatment of a wide class of batch optimization problems.

With the above discussion, and by the collection of Equations (1.1) through (1.8), the differential-algebraic optimization problem (DAOP) can be stated as:

$$\begin{array}{lll} \underset{\mathbf{x}(t),\mathbf{u}(t)}{min} & \Phi(\mathbf{x}(t), \mathbf{u}(t)) & t \in [t_0, t_f] \\ s.t.: & & \\ & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) & \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0 & \\ & \mathbf{c}(\mathbf{x}(t), \mathbf{u}(t)) = 0 & \\ & \mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U & \\ & \mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U & \end{array} \tag{1.9}$$

[2] In this thesis, the constraints such as: $\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U$ represent individual element-to-element bounds of the vector $\mathbf{x}$.

where, Φ represents the objective function, $\mathbf{g}$ and $\mathbf{c}$ are vectors containing functions in the algebraic constraints, $\mathbf{x}$ are the process states and $\mathbf{u}$ are the process inputs[3]. The solution to such a dynamic optimization problem is usually some trajectory for the decision variables of the optimization problem. Such decision variables could include set-points for the manipulated process variables (*i.e.*, input variables) or measured process variables (*i.e.*, output variables). It is then the task of process controllers to track or implement these optimal trajectories.

Dynamic optimization problems, as given in Problem (1.9), can be very difficult to solve analytically and numerical solution techniques are typically used. Conventional Nonlinear Programming (NLP) techniques cannot be directly applied to Problem (1.9) due to the presence of differential equality constraints. Optimal control methods, on the other hand, will deal with continuous control profiles but normally cannot handle general algebraic constraints (such as the ones represented by $\mathbf{g}$ or $\mathbf{c}$). Thus, special numerical techniques have to be employed in order to solve these problems. The next section presents the key characteristics of some of the techniques that have been reported in literature to solve problems of this type.

1.3 Solution Techniques

The DAOP stated in Equation (1.9) cannot be usually solved directly by typical nonlinear programming techniques or optimal control methods and analytical solutions are seldom available. Thus, when a DAOP (with or without uncertainty) is to be solved using a digital computer, it is typically necessary to first find a suitable numerical representation for candidate strategies. In this context, a variety of different representations and corresponding solution approaches have been suggested in the literature. The techniques most commonly employed to solve the differential-algebraic end-point optimization problem such as the one in DAOP (1.9) can be broadly classified into two classes: 1) methods that rely on some form of approximation of the differential equations in order to pose the problem as an NLP (*e.g.*, collocation based techniques, *etc.*); and 2) methods that embed a differential equation solver in the optimization strategy (*e.g.*, CVI, CVP, IDP, *etc.*).

Transformation of Problem (1.9) into an NLP form via discretization has been studied by a number of researchers (*e.g.*, Biegler [1984] ; Cuthrell and Biegler [1989]; Logsdon and Biegler [1989], and Villadsen and Michelsen [1978] , *etc.*). In these approaches to solving the DAOP, the state and manipulated variables are parameterized in terms of some set of basis functions (*e.g.*, simple polynomials [Villadsen and Michelsen, 1978] or Lagrange interpolation polynomials [Logsdon and Biegler, 1989] in t) and the optimization horizon (*i.e.*, $[t_0, t_f]$) is discretized in some fashion (*e.g.*, via orthogonal collocation [Cuthrell and Biegler, 1989]). As a result of discretizing the optimization time horizon, the number of

[3]This is a simplified version of the formulation given by Cuthrell and Biegler [1989] but contains all the characteristics necessary for the developments and discussions of this thesis.

equality constraints is inflated by the fineness of this discretization. Thus, increasing the accuracy of the solution via a finer discretization of the optimization horizon can drastically increase the computational requirements of this approach.

Examples of methods that embed differential equation solvers within the optimization technique include: *Control Vector Iteration* (CVI) [Ray, 1981], *Control Vector Parameterization* (CVP) [Hicks and Ray, 1971] and *Iterative Dynamic Programming* (IDP) [Luus, 1991; Dadebo and McAuley, 1995]). In CVI, the input vector $\mathbf{u}(t)$ is assumed to be piecewise constant and at each iteration of the optimization algorithm the state equations are integrated forward in time, then the results of the forward integration are used to integrate a set of adjoint equations backward in time to provide a correction to the current iterate for the input vector $\mathbf{u}_k(t)$. CVP eliminates the need for backward integration of the adjoint equations by expressing the input variable $\mathbf{u}(t)$ in terms of some predefined basis functions. IDP is a refinement of the general dynamic programming approach wherein at every iteration of the optimization algorithm: 1) a set of discrete input variable profiles $\mathbf{u}(t)$ are used to integrate the state equations forward in time; 2) the grid is refined during a backward pass; 3) the grid is contracted around the best input profile identified during the backward pass; and 4) the procedure is repeated until convergence. In each of these approaches, at least one integration of the state equations is required at every iteration of the optimization algorithm, which can be computationally expensive. Further, the accuracy of the solution depends upon the discretization chosen for the optimization time horizon.

Other Dynamic Programming [Bellman, 1957] approaches to such optimal control problems are usually based on the solution of the Hamilton-Jacobi-Bellman (HJB) equation, which is a set of nonlinear partial differential equations in the state variables. Many techniques have been applied to solve these equations. Lukes [1969] and Al'Brekht [1961] proposed perturbation methods to sequentially solve equations of increasing complexity. Lu [1993] proposed cost function transformation techniques that lead to simplifications of the HJB equation. Dolcetta [1983] and other researchers have considered the solution of the HJB equation using standard numerical techniques like finite element and finite difference methods. In a related approach, Beard [1998] has considered the solution of the HJB equations by successive Galerkin approximations. The resulting method is simple and can be used to generate approximate solutions that are optimal over a set of user-defined basis functions. Unfortunately, like other methods based on the HJB equation, and despite significant developments in the field of optimal control theory, the solution of this equation remains a formidable task that limits the use of nonlinear optimal control methods in practice.

1.4 Comparative Discussion

In all the methods discussed, the algorithmic solution procedures for DAOPs are parameterizations in one form or another. Thus, the problem complexity can be reduced enormously if it is known or assumed *a priori* that the class of candidate profiles can be restricted to sim-

ple parameterizations without losing significant optimization potential. In this framework, the issue of accuracy of the problem formulations chosen by different algorithms becomes important. Also, for the practical application of any of these techniques, the issues of solution convergence and computation time need to be analyzed. These issues become even more important for applications that need to be optimized in real-time or on-line.

The CVI approach converges relatively well for nonsingular problems, especially when used with a more sophisticated optimization technique than steepest descent, and yields results of high accuracy due to its relatively fine discretization. However, the computational cost of using a reliable integrator may be high. For singular problems, CVI may take longer to converge, as the gradient of the Hamiltonian depends indirectly on the control. As CVI is based on a condition of optimality that is only necessary but not sufficient, the user needs to numerically verify the optimality by introducing small perturbations. Moreover, state constraints or a large number of end-point constraints can significantly slow down the computations, although input constraints can be enforced with high accuracy. IDP requires a certain amount of tuning as far as region contraction factors and discretization fineness are concerned. The method handles constraints on control variables easily, but is not directly suitable for handling equality or inequality terminal constraints. Modifications to the IDP method, such as use of penalty functions or a use of an outer loop to optimize the parameters have to be done for different problem types, like minimum time problems. Nevertheless, IDP has been shown to be accurate with good convergence properties for small examples. However, depending on the process nonlinearities and the "stiffness" of the optimization problem, relatively fine level discretization at each instant is required and can substantially increase the computational effort. CVP does not have any significant advantages compared to other approaches, except that for some problems it might be faster than IDP, with/without a trade-off in accuracy. The convergence properties of CVP for the case of feedback parameterization are better, but with an increase in problem complexity. The collocation method on finite elements [Biegler, 1984; Cuthrell and Biegler, 1989] is a good candidate for solving even complex problems. However, the method requires a certain level of expertise in choosing an adequate number of finite elements and collocation points. Moreover, for large systems it may result in a large number of equations and constraints, thus dramatically increasing the size of the decision variable vector. Also, the collocation and optimization approach converges to a local optima, especially when successive quadratic programming is used from a poor starting point.

A comparison of major characteristics of the commonly used solution techniques for DAOPs is presented in Table 1.1. Specifically, the methods are compared on the basis of the following: 1) whether the control profile and/or the dynamic equations have to be discretized; 2) whether the problem formulation requires integration of differential equations at every iteration; 3) the overall problem complexity in terms of the number of decisions to be made; and 4) the computational effort required by different methods to solve the same

size and type of problem.

Table 1.1: Comparison of Algorithms for Solution to DAOPs.

Algorithm	Discretization	Integration	Complexity	Computation
CVI	√	√	LOW	HIGH
IDP	√	√	MEDIUM	HIGH
CVP	√	√	LOW	HIGH
collocation on finite elements	√		HIGH	MEDIUM

Looking at Table 1.1, it is clear that three out of four existing methods require integration of differential equations at every iteration, which increases the computation time dramatically. Collocation based technique, on the other hand, approximates the differential equations and hence eliminates the need for integration at every iteration. Although, this takes care of some of the computation time issues, the problem complexity is increased because of the large number of decision variables introduced by approximations. Moreover, the NLP method used to solve the approximated problem may converge to a local optima if the initial conditions for all these decision variables are not chosen properly. Generally, numerical approaches can produce solutions that are arbitrarily close to the optimum via increasingly fine discretization of time grid used in the problem by: 1) increasing the order of the approximating polynomials and/or; 2) choosing appropriate basis functions, *etc.*. However, this usually increases the computational requirements for solving the same problem. This trade-off between solution accuracy and computational requirements is due to the differential equations contained in Problem (1.9). If these could be transformed to algebraic equations, then conventional algebraic techniques (*i.e.*, Linear Programming (LP), Quadratic Programming (QP) and Nonlinear Programming (NLP)) could be used to efficiently and accurately solve the problem. Such a transformation is possible for *flat* nonlinear control systems.

1.5 Thesis Contributions

The central theme of this thesis is the optimal control trajectory generation for finite-time nonlinear dynamic systems, or in other words the solution to finite-time dynamic optimization problems. Recognizing the inefficiencies associated with existing techniques to solve such problems, an algorithm is proposed that simplifies (in some cases) the solution strategy. In some cases, existing numerical methods for the solution to such problems guarantee convergence and accuracy. However, their main drawback is that the problem formulations are composed of many decision variables, especially for large nonlinear dynamic systems. In these cases, the solution computation time becomes a serious concern, and hence

none of these techniques may be implemented in real-time without some modifications (except for the simplest of problems).

The main contribution of this thesis is the transformation of the dynamic optimization problem into a form that deals with some of these issues effectively. Using the concept of *flatness* and *dynamic time scaling* from nonlinear systems theory, an algorithm is proposed that transforms Problem (1.9) into an algebraic optimization problem (*i.e.*, LP, QP or NLP) when no path constraints exist, and to a semi-infinite optimization problem in the presence of path and input constraints. In other words, the algorithm permits the transformation of differential equations into algebraic ones, hence eliminating the need for integration to solve the optimization problem. This addresses the important issue of computation time for these problems. Moreover, it is shown that the issue of problem complexity (*i.e.*, the number of decision variables) is also addressed properly in such a formulation. The proposed scheme is illustrated on several nonlinear dynamic optimization problems reported in the chemical engineering literature.

Another contribution is the preliminary investigation performed on the applicability of this algorithm in a real-time trajectory generation framework. Although the investigations are performed with the most simplifying assumptions, they still lay the groundwork for further research in this area. The results reported in Chapters 3 and 4 provide a sufficient motivation for a careful look at the trajectory generation problem in the framework of this algorithm.

1.6 Thesis Organization

A brief overview of some of the techniques used for the analysis of nonlinear systems is presented in Chapter 2. The chapter focuses on the relatively recent differential geometric approaches that have been successfully applied for the analysis, design and control of nonlinear dynamic systems. In particular, the concepts of *differential flatness* and its direct extension, *viz.*, *orbital flatness* for dynamic systems are introduced. The algorithm proposed in this thesis concerns itself with optimization of finite-time dynamic processes that may be represented as *flat* systems, and hence this chapter lays the groundwork for the following chapters.

Chapter 3 proposes and describes the *Normalized Dynamic Optimization* (NDO) algorithm. The algorithm uses the concepts discussed in Chapter 2, to transform the differential equations in the DAOP to equivalent algebraic ones. The application of the algorithm is illustrated on four examples of differing complexity. Simulations are carried out to study the validity of this algorithm to solve the optimization problems, as well as to analyze the improvement in computation time compared to other techniques. In this context, the optimization results obtained are compared with the ones reported by the application of other techniques and algorithms.

Chapter 4 attempts to put the algorithm in a real-time trajectory generation framework,

in view of the encouraging results (especially in terms of computation time) reported in Chapter 3. The chapter focuses on the investigation of the proposed algorithm in a dynamic real-time optimization (DRTO) framework described by various researchers (*e.g.*, Ruppen *et al.* [1998], Loeblein *et al.* [1999], *etc.*). In particular, by formulating real-time estimation and optimization problems using this algorithm and the most simplifying assumptions, the chapter provides a motivation for further research in the area of real-time trajectory generation using this algorithm.

Chapter 5 summarizes the main contributions of this thesis and identifies areas where further research would be the most beneficial. The chapter also identifies some issues with the use of this algorithm in an off-line and/or on-line optimization application, and suggests some directions for future research.

Chapter 2

Nonlinear Dynamic Systems

This one's tricky. You have to use imaginary numbers, like eleventeen... — Bill Watterson, "Calvin and Hobbes"

Many common chemical manufacturing processes present challenging control problems, including nonlinear dynamic behaviour (*e.g.*, high purity distillation columns, highly exothermic chemical reactions, pH neutralizations, batch reactors, *etc.*). These processes may be required to operate over a wide range of conditions due to large process upsets or setpoint changes [Bequette, 1991]. In spite of this knowledge, the common approach in process control has been to neglect these nonlinear effects by locally linearizing the nominal model around the operating conditions and then to apply linear theory to design linear controllers. The technique may work well for mildly nonlinear continuous processes; the error introduced by locally linearizing around the steady-state being small enough so that it can be rejected easily by a sufficiently robust linear regulator. However, for severely nonlinear chemical processes such as batch reactions, control based on linear models may exhibit extremely poor performance in terms of robustness. The difficulties in applying linear theory for nonlinear processes are aggravated in servo control problems such as start-up/shut-down of continuous processes or optimal trajectory generation for batch processes where there is no appropriate point for local linearization [Kravaris and Chung, 1987]. A major reason to rely on linear control in such cases then, is that the theory is particularly rich and a wide variety of algorithms and methods exist for control design. Though linear control may result in undesirable control system performance, the theory is relatively easy to understand and computationally simple to implement.

In the past decade, however, the control of nonlinear systems has received considerable attention in both academia and industry [Henson and Seborg, 1997]. The recent interest in the analysis and design of nonlinear control systems is due to several factors: 1) linear controllers usually perform poorly when applied to highly nonlinear systems or even to moderately nonlinear systems that operate over a wide range of conditions; 2) significant progress has been made in the development of model-based controller design strategies for

nonlinear systems. These techniques use the nonlinear model directly, without the need for local linearization about an operating point; and finally 3) the development of inexpensive and powerful computers that have made on-line implementation of these nonlinear model-based controllers feasible. A detailed discussion of nonlinear control theory and analysis tools can be found in Isidori [1989] and Nijmeijer and van der Schaft [1990]. In the process control area, Kravaris and Kantor [1990a;b] and Bequette [1991] provide notable papers which include a tutorial and a detailed review of nonlinear process control systems.

The chapter provides an overview of the recent differential geometric approaches, that have been successfully applied to the analysis of nonlinear dynamic systems. In particular, the techniques of feedback linearization are discussed. The first section defines the terms and techniques used in studying dynamic control systems in a mathematical framework. In this framework, the systems are separated as being linear or nonlinear and this definition of nonlinear systems is used in the following sections[1]. The next section provides a brief overview of the differential geometric approaches as they are applied to nonlinear systems. Here, a brief description of the commonly used methods of feedback linearization, *viz.*, state feedback linearization, input-output linearization and approximate linearization is presented. In the next section, ***differential flatness***, which defines a special type of feedback linearization and which forms the basis for the optimization strategy proposed in this thesis, is defined. An overview of a few algorithms which may be used to identify the ***flat outputs*** for specific classes of nonlinear systems is also given, with the more involved definitions and theorems forming Appendices A and B, respectively. To extend the class of ***differentially flat*** systems, a suitable time scaling may be chosen and a section following that describes ***orbital flatness*** for ***time scaled*** nonlinear control systems. The concepts of ***differential*** and ***orbital flatness***, both are illustrated on typical control examples. Towards the end of the chapter, these concepts are summarized and discussed with regard to their applicability in the following chapters. Throughout this chapter, in order to preserve continuity, the descriptions are brief and just provide a glimpse into the various existing techniques. The interested reader is encouraged to look at the relevant references, as well as Appendices A and B for a broader overview of any of these concepts.

2.1 Dynamic Control Systems

Most dynamic control systems (lumped parameter) can be mathematically described as systems of under-determined ordinary differential equations (ODEs):

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \tag{2.1}$$

[1] It should be noted that any control system is a dynamic system. In order that the definitions presented here relate themselves directly to the following chapters, the dynamic systems are presented as control systems, *i.e.* with manipulated variables u. The results, however, are equally applicable to dynamic systems that do not differentiate between the state and manipulated variables, *i.e.*, x and u.

where $t \in \Re$ is time, $\mathbf{x} \in M \subset \Re^n$ the states and $\mathbf{u} \in \mathcal{U}^p$ are the manipulated variables or inputs for this system, respectively. This system is then under-determined by p equations, which is also the number of inputs for this system. Usually, in control applications, not all states can be measured. Only a portion of the state information is available through a set of measured outputs:

$$\mathbf{y} = \mathbf{h}(\mathbf{x}, \mathbf{u}, t) \tag{2.2}$$

where $\mathbf{y} \in \Re^m$ $(m \leq n)$. The basic problem in control theory is to choose the control $\mathbf{u}$ in such a way that a certain desired behaviour is established. Considering chemical processes, for example, the most common problem is that of determination of a control scheme that would take the system from a given initial condition to a given final condition in some given time, or that some desired quantity gets optimized while satisfying some constraints. Two types of control are generally considered: 1) *open-loop control* where a function $\mathbf{u} = \mathbf{u}(t)$ is chosen (specifying $\mathbf{u} = \mathbf{u}(t)$ in Equation (2.1) reduces the problem to one for which the dynamics are determined); and 2) *closed-loop control* where the control function depends on the past and/or instantaneous values of the state, *i.e.*, $\mathbf{u} = \mathbf{u}(\mathbf{x})$. For the case: $m \leq n$ in Equation (2.2), the control problem is more difficult; considering that only m out of n states can be observed. In this case, a feedback of the form: $\mathbf{u} = \mathbf{u}(\mathbf{y}, t)$ is sought. This control structure will be discussed in Chapter 4 in the application of real-time optimization to batch process trajectories.

A classification of control systems distinguishes the following types. A control system is called *linear* if the system (2.1) can be represented in the form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.3}$$

More specifically, linear systems are called *time-invariant* or *time-varying* depending on whether the matrices $\mathbf{A}$ and $\mathbf{B}$ are constant matrices or functions of time, t, respectively. *Nonlinear* systems, on the other hand, have the general representation given by Equation (2.1) where $\mathbf{f}$ is a mapping $(\mathbf{x}, \mathbf{u}) \mapsto \mathbf{f}(\mathbf{x}, \mathbf{u})$. A specific class of these nonlinear systems, called control-affine systems are widely encountered in chemical engineering (*e.g.*, feed rate problems), and have the following form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{g}(\mathbf{x})\mathbf{u}. \tag{2.4}$$

where, the term $\mathbf{f}(\mathbf{x})$ is called the drift vector field since it develops independent of the input to the system. These nonlinear control-affine dynamic systems are the focus of this study and are analyzed in the following sections.

2.2 Differential Geometric Approaches

A number of papers have provided insight into operational problems created by nonlinearities in chemical processes [Ray, 1982; Uppal *et al.*, 1974; Fox *et al.*, 1984; Morari, 1983;

Sieder *et al.*, 1991]. Initial solutions consisted primarily of hardware modifications such as equal-percentage valves, square-root extractors or other nonlinear control elements to remove major nonlinearities [Shinskey, 1962; Jones *et al.*, 1963], some of which are still in use today. More recently, controller design methods that provide exact linearization of nonlinear models have been developed. Unlike conventional linearization via Taylor series expansion, these techniques produce linearized models that are independent of the operating point. This general approach has been used in several analysis and design methods (*e.g.*, Variable Transformations [Skogestad and Morari, 1988; Georgakis, 1986] and Internal Model Approaches /Garcia and Morari, 1982], *etc.*). Bequette [1991] provides an excellent review in the area of nonlinear process control and should be referred to for more details.

One of the important techniques that has enjoyed considerable attention in the study of nonlinear systems is that of differential geometry. An overview of geometrical methods for process control is given by Hunt *et al.* [1987]. Henson and Seborg review geometric control methods [1990] and present a general (unified) approach [1989] . Kravaris and Kantor [1990a;b] provide a tutorial for many of the details of differential-geometric-based control system design. McLellan *et al.* [1990] review error trajectory techniques and place them in the context of differential geometric approaches.

The problem of exact linearization of nonlinear systems by coordinate transformations and feedback has been an active area of research over the last 20 years [Isidori, 1989]. It is widely recognized, however, that linearizability is not a generic property of nonlinear control systems, meaning that the class of feedback linearizable nonlinear systems forms a subset of all nonlinear systems [Rouchon, 1993; Tchòn, 1994]. Moreover, no robust theory or results exist for their characterization, except for some special classes. However, for systems that are somehow identified to be feedback linearizable, a number of applications have appeared in the literature that demonstrate the usefulness of this concept (*e.g.*, Henson and Seborg [1997], Rathinam and Sluis [1995], *etc.*).

State feedback transformations are characterized by a special class of transformations given by:

$$\{t, \mathbf{x}, \mathbf{u}\} \longmapsto \{t, \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\psi}(\mathbf{x}, \mathbf{u})\} \tag{2.5}$$

where, the states and the manipulated variables in the original coordinates: $\mathbf{x}$ and $\mathbf{u}$,are mapped to a different set of coordinates: $\boldsymbol{\phi}(\mathbf{x})$ and $\boldsymbol{\psi}(\mathbf{x}, \mathbf{u})$, respectively. The time, t, is considered to be a given physical entity and hence is not transformed in the usual case. More general feedback transformations have also been considered where equivalences of nonlinear systems under transformation of time scale have been shown to be useful [Guay, 1999; Martin, 1992]. This case: $\{t, \mathbf{x}, \mathbf{u}\} \longmapsto \{\tau(\mathbf{x}, t), \boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\psi}(\mathbf{x}, \mathbf{u})\}$ will be considered in a following section of this chapter, where orbital flatness is discussed.

In engineering practice as well as in analysis, the control system given by Equation (2.1) is often modified by adding a *feedback*. This is illustrated in Figure 2.1. The block marked $\mathbf{f}$ refers to the control system that has the states $\mathbf{x}$ as its outputs. The block marked $\boldsymbol{\phi}$ can be

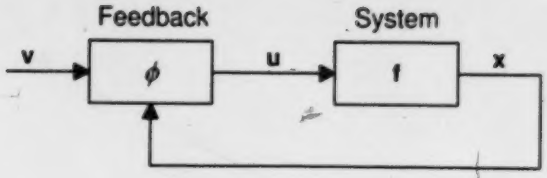


Figure 2.1: Depiction of Feedback of Control Systems.

thought of as the feedback or an operator that maps a set of reference inputs $\mathbf{v} = (v_1, ..., v_p)$ and the states $\mathbf{x}$ to the nominal controls: $\mathbf{u} = (u_1, ..., u_p)$. This feedback is called a *static state feedback* when the operator $\boldsymbol{\phi}$ is given by a map $\boldsymbol{\gamma} : \Re^{n+p+1} \rightarrow \Re^p$ in the form:

$$\mathbf{u}(t) = \boldsymbol{\gamma}(t, \mathbf{x}(t), \mathbf{v}(t)) \tag{2.6}$$

The feedback itself may involve some dynamics that can be represented as:

$$\begin{aligned} \dot{\mathbf{z}}(t) &= \boldsymbol{\alpha}(t, \mathbf{x}(t), \mathbf{z}(t), \mathbf{v}(t)) \\ \mathbf{u}(t) &= \boldsymbol{\beta}(t, \mathbf{x}(t), \mathbf{z}(t), \mathbf{v}(t)) \end{aligned} \tag{2.7}$$

where $\mathbf{z} = \{z_1, ..., z_m\}$ are called the new states. The composite system may be thought of as a control system with $\mathbf{v}$ as its p inputs and $\{\mathbf{x}, \mathbf{z}\}$ as its $n + m$ states. This type of feedback is called a *dynamic state feedback*. When $m = 0$ the dynamic feedback becomes a static feedback and hence static feedback is just a special case of dynamic feedback.

After a static feedback and a possible nonlinear transformation of the state variables some control systems may be expressed in the linear form:

$$\dot{\boldsymbol{\xi}} = \mathbf{A}\boldsymbol{\xi} + \mathbf{B}\mathbf{v} \tag{2.8}$$

where $\boldsymbol{\xi} = \boldsymbol{\delta}(\mathbf{x})$ are the new coordinates for the states. Such systems are said to be *feedback linearizable via static state feedback* and have been completely classified in literature [Isidori, 1989]. Systems that are not static state feedback linearizable may still be *dynamic state feedback linearizable* in the sense that after a dynamic feedback and a diffeomorphism of the states $\boldsymbol{\xi} = \boldsymbol{\delta}(\mathbf{x}, \mathbf{z})$ they take the linear form of Equation (2.8). Classification of dynamic feedback linearizability is still an open problem though classification results exist for special classes of systems. For instance in the case of $p = 1$ dynamic feedback linearizability has been shown to be equivalent to static feedback linearizability [Shadwick, 1990].

2.2.1 State Feedback Linearization

A state feedback compensator law, such as the one in (2.6) or (2.7), is found so that after a coordinate transformation, $\boldsymbol{\xi} = \mathbf{T}_1(\mathbf{x})$ and $\mathbf{v} = \mathbf{T}_2(\mathbf{x}, \mathbf{u})$, the relationship between the reference input, $\mathbf{v}$, and the dynamic state equation is linear [Hunt *et al.*, 1983]. The system

can then be equivalently written as:

$$\dot{\boldsymbol{\xi}} = \mathbf{A}\boldsymbol{\xi} + \mathbf{B}\mathbf{v}.$$

Brockett [1978] and Jakubczyk and Respondek [1980] provide algorithms for feedback linearization of single-input and multi-input systems respectively. Charlet *et al.* [1989] provide a detailed description of dynamic feedback linearization. A major disadvantage with this method is that a set of partial differential equations must be solved to determine the variable transformations and an analytical solution is only available in special cases.

2.2.2 Input-Output Linearization

In this type of linearization, a state compensator feedback law is found such that the relationship between the reference input, v, and the output, y, is linear [Isidori, 1989; Kravaris and Chung, 1987]. Any linear design technique can then be used for the controller. It was shown that for control-affine nonlinear control systems, the state feedback compensator law under PI control results in a closed-loop transfer function:

$$y(s) = \frac{1}{(\tau s + 1)^r} y_{sp}(s). \tag{2.9}$$

where r is the relative order. A number of important chemical processes have a relative order of 1, thus yielding a first-order closed-loop response using this approach. A disadvantage of this method is that it can only be used for minimum-phase systems, though some extensions to non-minimum phase systems have been proposed [Wright and Kravaris, 1990].

2.2.3 Approximate Linearization

The problem of approximate linearization was first treated by Krener [1984] . The technique consists of finding feedback and state space transformations which are polynomials of order κ in the states and $\kappa - 1$ in the inputs, such that an order κ approximate linear system is obtained

$$\dot{\boldsymbol{\xi}} = \mathbf{A}\boldsymbol{\xi} + \mathbf{B}\mathbf{v} + \boldsymbol{\vartheta}(\mathbf{x}, \mathbf{u})^{\kappa+1}$$

where $\boldsymbol{\xi} = \mathbf{T}_1(x)$ and $\mathbf{v} = \mathbf{T}_2(\mathbf{x}, \mathbf{u})$ are the transformed states and inputs respectively. Guzzella and Isidori [1993] and Hunt and Turi [1993] extended this approach to multi-input systems using simplified approaches. Using these techniques, it is relatively easy to find coordinate transformations and feedbacks which linearize a system. Although not an issue in most of these developments, such techniques are approximate and validity of the approximations must be checked.

The above linearization approaches, though useful in some situations, impose serious restrictions on the structure of the nonlinear control system that are seldom met in practice. To overcome this problem, new approaches to the problem have been presented in

control literature that have led to new generalizations of the standard feedback linearization techniques. New definitions of feedback linearizability have emerged to relax some of the restrictions originally imposed by the requirement of the state-feedback case. Charlet *et al.* [1989] introduced the concept of dynamic feedback linearization and derived conditions for linearizability under dynamic state feedback transformations. It was shown that the problem of dynamic feedback linearization amounts to the design of a precompensator that can be coupled with the original process model to yield a linearizable structure. Fliess *et al.* [1995] generalized the dynamic feedback linearizability using the concept of *differential flatness*. This important concept equates the dynamic feedback linearizability of nonlinear systems by endogenous feedback to the existence of *flat* (or *linearizing*) *outputs*. These *flat outputs* are such that they summarize the entire dynamics of the process (*i.e.*, every state and input of the system can be expressed as some function of the *linearizing outputs* and a finite number of their time derivatives). A detailed description of this concept follows.

2.3 Differential Flatness

Differential flatness was originally introduced by Martin [1992] and studied further by Fliess *et al.* [1995] and van Nieuwstadt *et al.* [1995]. It refers to the existence of so-called *flat* or linearizing outputs that summarize the dynamics of a nonlinear system. It is closely related to the general ability to linearize a nonlinear system by an appropriate choice of endogenous dynamic feedback. An endogenous feedback is a dynamic feedback of the form given in Equation (2.7) with the added requirement that $\mathbf{z}$ and $\mathbf{v}$ be uniquely determined as functions of $t, \mathbf{x}, \mathbf{u}$ and a finite number of their derivatives. They have the useful property that there is a one-to-one mapping between trajectories in this *flat output* space and trajectories in state space. Trajectories can thus be planned in the lower dimensional *flat output* space and then lifted to the state and input space, through an algebraic mapping. A variety of examples have been shown to be *differentially* or *approximately flat*. These examples include chemical systems (*e.g.*, Rothfuss [1996]), mechanical systems (*e.g.*, Rouchon [1993] and Murray [1995]) and aircraft systems (*e.g.*, Martin *et al.* [1994]). This thesis also exploits *differential flatness*, but with an emphasis on the solution to optimal control problems as well as to the real-time trajectory generation for nonlinear *differentially* or *orbitally flat* chemical processes. *Orbital flatness*, a more general concept for nonlinear systems, was introduced by Fliess [1993]. This property refers to the linearizability of systems subject to an endogenous state feedback transformation and a state dependent time scaling transformation. It was shown by Fliess *et al.* [1997] and Guay [1999] that this concept can be viewed as a generalization of the *differential flatness* property for weakly locally accessible nonlinear systems.

Differential flatness is a concept that applies to under-determined system of ordinary differential equations (ODEs). As discussed earlier, a control system with n states and p inputs can be written as a system of under-determined differential equations represented by

Equation (2.1). *Differential flatness* for this system is then defined as follows:

Definition 1 (Rathinam, 1995) *The system given by equation (2.1) is said to be differentially flat or simply flat if there exist variables* $\mathbf{y} = \{y_1, ..., y_p\}$ *given by an equation of the form:*

$$\mathbf{y} = \mathbf{h}(t, \mathbf{x}, \mathbf{x}^{(1)}, ..., \mathbf{x}^{(m)}, \mathbf{u}, \mathbf{u}^{(1)}, ..., \mathbf{u}^{(m)}) \tag{2.10}$$

such that the original variables $\mathbf{x}$ *and* $\mathbf{u}$ *may be recovered from* $\mathbf{y}$ *(locally) by equations of the form:*

$$\begin{aligned} \mathbf{x} &= \mathbf{g}_1(t, \mathbf{y}, \mathbf{y}^{(1)}, ..., \mathbf{y}^{(l)}) \\ \mathbf{u} &= \mathbf{g}_2(t, \mathbf{y}, \mathbf{y}^{(1)}, ..., \mathbf{y}^{(l)}) \end{aligned} \tag{2.11}$$

The variables $\mathbf{y} = (y_1, ..., y_p)$*, are referred to as the flat outputs. Flatness may be regarded as a (local) bijective correspondence between the solutions* $\mathbf{x}(t)$ *of Equation (2.1) and arbitrary curves* $\mathbf{y}(t)$ *in* $\Re^p$ *that is given by the maps* $\mathbf{h}$, $\mathbf{g}_1$ *and* $\mathbf{g}_2$ *of the Equations (2.10) and (2.11).*

It must be noted that not all nonlinear systems possess *flat outputs*. In the following example, the property of *differential flatness* is illustrated on a 4-state, 2-input example.

2.3.1 Illustrative Example

Consider the control-affine system of Charlet *et al.* [1989,1991]

$$\begin{aligned} \dot{x}_1 &= x_2 + x_3 u_2 \\ \dot{x}_2 &= x_3 + x_1 u_2 \\ \dot{x}_3 &= u_1 + x_2 u_2 \\ \dot{x}_4 &= u_2. \end{aligned} \tag{2.12}$$

This control system with x_1, x_2, x_3 and x_4 as its states and u_1 and u_2 as its inputs can be considered as a system of ODEs which is under-determined by two equations. For control, when the two inputs u_1 and u_2 are set to some arbitrary functions of time (or states), a fully determined system of ODEs is obtained, which can be solved to find the instantaneous value of the states. In general, from a mathematical point of view, this system of ODEs may be solved by specifying *any* two variables. The set of solutions of this resulting system may then depend on some initial conditions. For instance specifying u_1 and u_2, a system whose solution depends on four constants, which are initial conditions for x_1, x_2, x_3 and x_4 is obtained. Thus the entire set of solutions $\{x_1(t), x_2(t), x_3(t), x_4(t), u_1(t), u_2(t)\}$ can be considered to be parameterized by two arbitrary functions (which specify $u_1(t)$ and $u_2(t)$) and four arbitrary constants (which specify the initial conditions for $x_1(t), x_2(t), x_3(t)$ and $x_4(t)$).

Differential flatness of systems such as the one in Equation (2.12) permits a choice of the free variables (the ones that are assigned to arbitrary functions) such that no constants are required to parameterize the solution set. For the system in Equation (2.12), no pair of

variables from the set $\{x_1, x_2, x_3, x_4, u_1, u_2\}$ would achieve this property. However, *pseudo outputs:* y_1 and y_2, defined as functions of the original variables such as the following:

$$\begin{aligned} y_1 &= x_4 \\ y_2 &= x_2 u_2 - x_1 \end{aligned} \tag{2.13}$$

would parameterize the entire set of solutions without the need for additional constants. This is evident from the solution of the variable set $\{x_1, x_2, x_3, x_4, u_1, u_2\}$ as functions of y_1 and y_2 defined in Equation (2.13). The resulting expressions are given by[2]:

$$\begin{aligned} x_1 &= \dot{y}_1 \left(\frac{\dot{y}_2 + \dot{y}_1^2 y_2}{\ddot{y}_1 + \dot{y}_1^3 - 1} \right) - y_2 \\ x_2 &= \frac{\dot{y}_2 + \dot{y}_1^2 y_2}{\ddot{y}_1 + \dot{y}_1^3 - 1} \\ x_3 &= \frac{\dot{x}_1 - x_2}{u_2} = f_1(\bar{y}_1, \bar{y}_2) \\ x_4 &= y_1 \\ u_1 &= \dot{x}_3 - x_2 u_2 = f_2(\bar{y}_1, \bar{y}_2) \\ u_2 &= \dot{y}_1 \end{aligned} \tag{2.14}$$

The variables y_1 and y_2 in the above example are called *flat outputs*. In this example the *flat outputs* were functions of the original variables x_1, x_2, x_3, x_4, u_1 and u_2. In general these can be functions of finitely many derivatives of the original variables.

Flatness of the dynamics given in Equation (2.12), thus enables the representation of the system in terms of two pseudo-variables called *flat outputs*. Once these *flat outputs* are assigned to arbitrary functions of time, the system is fully defined, without the need for any initial conditions on the variables. Further, the trajectories for the system states can be calculated without the need for integration of differential equations that would have been required if the input variables: u_1 and u_2 were defined as arbitrary functions of time. This powerful concept is illustrated in the next chapter, where optimal state trajectories are determined using a suitable parameterization of the *flat outputs* in time.

2.4 Orbital Flatness

For the treatment of so-called *non-flat* systems, Fliess *et al.* [1993] have considered a more general definition of linearizability called *orbital flatness*. Roughly speaking, *orbital flatness* refers to *differential flatness* of a system that has been transformed to some other time coordinate. Time coordinate transformations or time scalings, hence form an integral part of *orbitally flat* systems. Respondek [1998] developed a set of necessary and sufficient conditions for the *orbital feedback linearization* of single-input systems. The *orbital flatness* of a simple reactor model due to Kravaris and Chung [1987] was demonstrated in Guay [1999].

[2] In the expressions for x_3 and u_1, only the functional dependence is shown because the resulting expressions are quite large. The outputs y_1, y_2 and their derivatives up to the fourth order have been collected in $\bar{y}_1$ and $\bar{y}_2$ respectively, i.e. $\bar{y}_1 = \{y_1, \dot{y}_1, \ldots, \ddot{\ddot{y}}_1\}$ and $\bar{y}_2 = \{y_2, \dot{y}_2, \ldots, \ddot{\ddot{y}}_2\}$.

In this thesis, the algorithm of Guay [1999] is used to identify the time-scale transformations (if possible) that make the *non-flat* systems *orbitally flat*. The discussion on *orbital flatness* in this section is organized as follows. First, an overview of time scaling transformations for nonlinear systems is given followed by a definition of *orbital flatness*. The next section illustrates the algorithm of Guay [1999] on a batch non-isothermal reactor and identifies the time scaling and the *flat output* for this system.

2.4.1 Time Scaling Transformations

In the analysis of any continuous time system, the time scale t is assumed to be specified and independent. For many analysis tasks, it is often cumbersome to restrict dynamics to evolve with respect to respect to the actual time t. In a number of approaches, a new time scale τ is introduced to facilitate the analysis. This is the case in most singular perturbation methods [Kokotovic, 1986] where the time scale of the slow dynamics are adjusted to facilitate stability analysis. Hollerbach [1984] used a new time scale r for trajectory planning of robots, but his work is restricted to robots and r is introduced as a function of t, *i.e.*, $r(t)$. Sampei and Furuta [1986] defined a new time scale τ using a time scaling function $0 < s(\mathbf{x}) < \infty$ as $dt/d\tau = s(\mathbf{x})$. A time scaling of this type is convenient for system analysis because the state differential equation can be easily rewritten in the new time scale τ and ordinary methods for analysis can be applied for the rewritten system.

Consider a smooth nonlinear control system of the form:

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{2.15}$$

where $\mathbf{x} \in M \subset \Re^n$, $\mathbf{u} \in \mathcal{U}^p$. A general time scaling transformation τ can be defined as:

$$\frac{dt}{d\tau} \equiv s(\mathbf{x}, \mathbf{u}), \quad \tau|_{t_0} = \tau_0, \tag{2.16}$$

where t is the actual time. The function $s(\mathbf{x}, \mathbf{u})$ is assumed to be smooth such that $0 < s(x, u) < \infty$ and is called a *time scaling function*. The system in Equation (2.15) can then be rewritten in the new time scale τ as:

$$\frac{d\mathbf{x}}{d\tau} \equiv \mathbf{g}(\mathbf{x}, \mathbf{u}) = s(\mathbf{x}, \mathbf{u})\mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{2.17}$$

In Equation (2.17), $g(\mathbf{x}, \mathbf{u})$ is well-defined because $s(\mathbf{x}, \mathbf{u}) \neq 0$. The restriction $0 < s(\mathbf{x}, \mathbf{u}) < \infty$ is very important to ensure that new time τ is a strictly monotone, increasing function with respect to the actual time t. This guarantees that stability and the state trajectory of the system is preserved by the transformation of time scale. Moreover, the time scaling function $s(\mathbf{x}, \mathbf{u})$ is required to be continuous with respect to $\mathbf{x}$ and $\mathbf{u}$ in order to preserve the smoothness of vector fields.

For single-input systems, necessary and sufficient conditions for the feedback linearization of control-affine nonlinear systems under state-dependent time scaling functions were first derived by Sampei and Furuta [1986]. These conditions were later refined by Respondek

[1998] and Guay [1999]. The algorithm presented by Guay [1999] identifies the linearizing outputs for single-input control-affine nonlinear systems using an exterior calculus setting, which simplifies some aspects of the computations required.

Orbital flatness, the more general concept of *flatness*, that allows time scaling transformations such as (2.16), is defined next.

Definition 2 *The nonlinear control system given by Equation (2.17) is said to be orbitally flat if there exist variables* $y(\tau) = \{y_1(\tau), ..., y_p(\tau)\}$ *given by an equation of the form:*

$$\begin{aligned} \mathbf{y}(\tau) &= \mathbf{h}(\tau, \bar{\mathbf{x}}_m, \bar{\mathbf{u}}_m) \\ \bar{\mathbf{x}}_m &= \left\{ \mathbf{x}, \frac{d\mathbf{x}}{d\tau}, \frac{d^2\mathbf{x}}{d\tau^2}, \ldots, \frac{d^m\mathbf{x}}{d\tau^m} \right\} \\ \bar{\mathbf{u}}_m &= \left\{ \mathbf{u}, \frac{d\mathbf{u}}{d\tau}, \frac{d^2\mathbf{u}}{d\tau^2}, \ldots, \frac{d^m\mathbf{u}}{d\tau^m} \right\} \end{aligned} \tag{2.18}$$

such that the original variables $\mathbf{x}$ *and* $\mathbf{u}$ *may be recovered from* $\mathbf{y}(\tau)$ *(locally) by equations of the form:*

$$\begin{aligned} \mathbf{x} &= \mathbf{g}_1(\tau, \bar{\mathbf{y}}_l) \\ \mathbf{u} &= \mathbf{g}_2(\tau, \bar{\mathbf{y}}_l) \\ \bar{\mathbf{y}}_l &= \left\{ \mathbf{y}, \frac{d\mathbf{y}}{d\tau}, \frac{d^2\mathbf{y}}{d\tau^2}, \ldots, \frac{d^l\mathbf{y}}{d\tau^l} \right\} \end{aligned} \tag{2.19}$$

The variables $\mathbf{y}(\tau) = \{y_1(\tau), ..., y_p(\tau)\}$*, are referred to as the flat outputs. The new time scale* τ *is obtained as a solution to the differential equation (2.16).*

A system that is *orbitally flat* need not be *flat* in the actual time t. It is meaningful to consider *orbital flatness* because a transformation of time scale preserves stability. Also, it is useful because *differential flatness* can then be extended to an even greater class of nonlinear systems, thus making the analysis tasks for these systems easier. In this thesis, *differential flatness* in the new time scale is shown to facilitate the solution of differential-algebraic optimization problems.

2.4.2 Illustrative Example

Consider the dynamics of the batch non-isothermal chemical reactor from Kravaris and Chung [1987], in which the following consecutive reaction takes place:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C \tag{2.20}$$

It is assumed that $A \xrightarrow{k_1} B$ has second-order kinetics whereas $B \xrightarrow{k_2} C$ has first-order kinetics. The system dynamics are then modeled as:

$$\begin{aligned} \dot{c}_A &= -k_1(T)c_A^2 \\ \dot{c}_B &= k_1(T)c_A^2 - k_2(T)c_B \\ \dot{T} &= \gamma_1 k_1(T)c_A^2 - \gamma_2 k_2(T)c_B + (a_1 + a_2 T) + (b_1 + b_2 T)u \end{aligned} \tag{2.21}$$

where c_A and c_B are the molar concentrations of species A and B, T is the reactor temperature, u is the manipulated variable which is a dimensionless quantity related to the heating and cooling capacity of the reactor jacket, k_1 and k_2 are the reaction rate constants assumed to be functions of temperature by an Arrhenius type relation. The variables $\gamma_1, \gamma_2, a_1, a_2, b_1$ and b_2 are treated as constants. As discussed in Kravaris and Chung [1987], this system is not feedback linearizable. However, as shown in Guay [1999] , this system is *orbitally feedback linearizable* (*orbitally flat*) with the following choice of the time scaling transformation:

$$\frac{dt}{d\tau} = \frac{c_B}{k_1(T)c_A^2} \tag{2.22}$$

and c_A as the *flat output*. For illustration purposes, the system dynamics (2.21) are written in the new time scale as:

$$\frac{dc_A}{d\tau} = -c_B \tag{2.23}$$

$$\frac{dc_B}{d\tau} = c_B - \frac{k_2(T)c_B^2}{k_1(T)c_A^2} \tag{2.24}$$

$$\frac{dT}{d\tau} = \gamma_1 c_B - \gamma_2 \frac{k_2(T)c_B^2}{k_1(T)c_A^2} + \frac{c_B}{k_1(T)c_A^2}\left[(a_1 + a_2 T) + (b_1 + b_2 T)u\right] \tag{2.25}$$

Now, choosing the flat output as:

$$y(\tau) = c_A \tag{2.26}$$

the system states c_A, c_B, T and the system input u are uniquely described as functions of this *flat output* and a finite number of its derivatives. Using Equations (2.23) and (2.26), it follows:

$$c_B = -\dot{y}(\tau) \tag{2.27}$$

Now substituting Equations (2.26) and (2.27) in Equation (2.24) yields an expression for the temperature[3], T :

$$T = \alpha(y(\tau), \dot{y}(\tau), \ddot{y}(\tau)) \tag{2.28}$$

The function for u can then be solved for by the substitution of Equations (2.26), (2.27) and (2.28) into Equation (2.25) which yields:

$$u = \beta(y(\tau), \dot{y}(\tau), \ddot{y}(\tau)) \tag{2.29}$$

Thus c_A indeed forms an *orbitally flat output* for the system (2.21).

[3]Here, and in the expression for u, only the functional dependence is shown because the resulting expressions are quite large.

2.5 Characterization of Flatness

In differential algebra, a system is viewed as a differential field generated by a set of variables (states and inputs). The system is said to be *differentially flat* if a set of variables, called the *flat outputs* can be found, such that the system is (non-differentially) algebraic over the differential field generated by the set of *flat outputs*. *Differential flatness* can also be characterized using tools from exterior calculus[4]. In the beginning of this century, the French geometer E. Cartan developed a set of powerful tools for the study of equivalence of systems of differential equations [Cartan, 1953]. It was pointed out in the paper that equivalence need not be restricted to systems of equal dimensions. It was discussed that a system can be prolonged to a bigger system on a bigger manifold, and equivalence between these prolongations can be studied. Two systems that have equivalent prolongations were called absolutely equivalent. *Differentially flat* systems were then interpreted as being those systems which are absolutely equivalent to the trivial system, *i.e.*, having no dynamic constraints on the free variables [Sluis, 1992].

Nieuwstadt *et al.* [1994] interpreted *flatness* in an exterior calculus setting, using the tools offered by Cartan [1953] . In this setting, a fully nonlinear system can be seen to be defined on a manifold of dimension $1 + n + p$ (n and p are the number of states and inputs, respectively) and defines in particular a Pfaffian system. For system (2.1) it would be generated by

$$dx_1 - f_1(x, u, t)dt, \ldots, dx_n - f_n(x, u, t)dt$$

and the dynamics of the control system will then be described by integral curves. This approach gives an explicit treatment of time dependence. A complete characterization of *flatness* for single-input systems was also given in Shadwick [1990].

Differential flatness of a nonlinear system is very difficult to prove in general. There are no necessary and sufficient conditions for the existence of linearizing outputs. As a result, one can only prove *differential flatness* when a carefully chosen set, or a subset, of linearizing outputs can be found. In this direction, for single input systems, Shadwick [1990] showed that *flatness* is equivalent to static state-feedback linearizability. In this special case, it is relatively easy to uncover the linearizing outputs by using the well-known GS algorithm [Gardner and Shadwick, 1992]. For systems where the number of states exceeds the number of inputs by one, Charlet *et al.* [1991] demonstrated that dynamic feedback linearizability (*i.e.*, *flatness*) is assured if the system is locally strongly accessible. In Martin and Rouchon [1995] , it was demonstrated that any driftless nonlinear system with n states and $n - 2$ inputs is *flat*.

In general, the multi-input case is difficult because one has to prove, at least partially, the dynamic feedback linearizability of the nonlinear system by endogenous feedback. To avoid

[4] An overview of exterior calculus including the definition of the commonly used terms (including the ones in the following discussion) is provided in Appendix A.

this problem, Rathinam and Sluis [1995] showed that, for *flat* systems, the computation of *flat outputs* can be reduced to a one-dimensional problem. By a judicious parameterization of the state space, multi-input integrability condition can be reduced to a single integrability condition similar to the single-input situation. Application of the GS algorithm provides the remaining linearizing outputs. Unfortunately, this elegant result can be difficult to apply in situations where a suitable subset of *flat outputs* cannot be found. An alternative approach was developed by Guay *et al.* [1997] who considered the problem of dynamic linearization by endogenous dynamic feedback. Using an exterior calculus approach, they developed a set of necessary and sufficient conditions for dynamic feedback linearizability applicable to a large class of linearizable systems with arbitrary precompensators. Assuming a specific structure for the dynamic precompensator, these conditions yield an algorithm to identify linearizing outputs of control-affine systems.

In general, the conditions for linearizability by static or dynamic state feedback can be quite restrictive. These restrictions can be relaxed somewhat by considering state-dependent time scaling transformations along with a state feedback and a local change of state coordinates. In any case, a complete characterization of *flatness* for multi-input systems is not yet available; algorithms that aid in the identification of *flat outputs* exist for specific cases only. It is clear from the above discussion that the characterization of *flatness*, although difficult, is important and has been studied by various researchers. Although none of the techniques promise a guaranteed solution to the characterization, a variety of results have been reported, some of which are: 1) Rathinam and Sluis [1995] Algorithm based on the reduction of the multi-input problem to a single-input one; 2) Gardner and Shadwick [1992] (GS) Algorithm which defines the necessary and sufficient conditions for linearizability of nonlinear systems; 3) Guay *et al.* [1997] Algorithm that provides the necessary and sufficient conditions for dynamic feedback linearizability by endogenous feedback for control-affine nonlinear systems; and 4) Guay [1999] Algorithm for orbital feedback linearization of single-input control affine systems. In this thesis, one of the latter two algorithms, depending on the problem, is used to identify the *flat outputs* for the nonlinear control systems. The important details including the relevant theorems and references for the above algorithms are given in Appendix B. In the following discussion, a brief overview of these techniques is presented.

2.5.1 Reduction of System Codimension

System codimension is essentially one plus the number of equations by which the system is under-determined (*i.e.*, for control systems the system codimension will be $1+p$, where p is the number of inputs). The method of reduction of system codimension, generally speaking, refers to the technique of guessing all but one independent variable, thus ending up with a system of codimension 2. This technique was proposed by Rathinam and Sluis [1995] considering the fact that a complete characterization of *flatness* for single-input systems

(codimension 2 systems) is known. For systems with codimension greater than two, no complete characterization of *flatness* exists, except for some verifiable necessary conditions. A scheme was proposed that involved making a guess for all but one of the *flat outputs* (*i.e.*, setting all but one *flat outputs* to arbitrary functions $f^j(t)$), after which the problem reduces to a single-input problem. The last *flat output* was then characterized using existing algorithms. With the aid of the theory of absolute equivalence, the validity of the method was demonstrated [Rathinam and Sluis, 1995]. The method was also illustrated on two examples, one of which was non-trivial and was not known to be *flat* before. The algorithm and the relevant details are summarized in Appendix B. The major disadvantage with this approach is the fact that a suitable structure for all but one *flat outputs* has to be guessed, which might not be possible for most nonlinear systems.

2.5.2 The Gardner and Shadwick (GS) Algorithm

The *Brunovsky normal form* is the standard linear representation of controllable linearizable systems. In many applications, including that of characterizing *flat outputs*, it is necessary to compute the explicit formula for the feedback transformations which take a linearizable nonlinear system into its standard *Brunovsky normal form*. The GS algorithm [Gardner and Shadwick, 1992] utilizes the minimum number of integrations required for the exact linearization of nonlinear systems to *Brunovsky normal form* under nonlinear feedback and hence provides a necessary and sufficient condition for feedback linearizability under any kind of dynamic state feedback. The algorithm thus adapts itself well to the problem of characterization of *differential flatness* (which is just a special case of dynamic feedback linearization). The tools involved in the algorithm [Gardner and Shadwick, 1992] are based on classical constructions appearing in the theory of exterior differential systems, and is algebraically and computationally attractive. The algorithm, particularly the necessary and sufficient condition for linearizability of nonlinear systems utilized in the development of the approach, is summarized in Appendix B. The following two algorithms build on this GS algorithm to adapt themselves well to the problem of exact linearization by endogenous dynamic feedback (*differential flatness*).

2.5.3 Linearization by Endogenous Dynamic Feedback

Guay *et al.* [1997] present a necessary and sufficient condition for dynamic feedback linearizability by endogenous feedback of control-affine nonlinear systems. In other words, the algorithm provides a test for *differential flatness* (*i.e.*, it determines whether a system is *flat* or not) and also enables the identification of the *flat output*, if the system is indeed *flat*. The necessary and sufficient condition is based on a modified derived flag of the Pfaffian system that takes into account the presence of a precompensator. As for the GS algorithm, it is shown that the generators associated with this modified derived flag must satisfy a number of congruences. The congruences ensure that the conditions for the GS algorithm

are satisfied for the prolongated system and thus provide an algorithm to calculate the required feedback and state space transformations. It was shown how linearizing outputs can be identified by studying the derived flag of the Pfaffian system associated with a nonlinear system [Guay *et al.*, 1997]. Appendix B presents a more detailed description of the algorithm and the terminology.

2.5.4 An Algorithm for Orbital Feedback Linearization

For cases where linearizability conditions cannot be met, Fliess *et al.* [1993] considered a more general definition of linearizability called *orbital flatness*. In this definition of *flatness*, some of the conditions of dynamic feedback linearization by endogenous feedback have been relaxed. Respondek [1998] developed a set of necessary and sufficient conditions for the orbital feedback linearization of single-input systems. Guay [1999] considered the problem of orbital feedback linearization using an exterior calculus approach. A necessary and sufficient condition for *orbital flatness* for single-input control-affine systems was presented. A simple algorithm computes the state-dependent time scaling that yields state-feedback linearizable systems [Guay, 1999]. This algorithm and the relevant details are presented in Appendix B.

2.6 From Flatness to Trajectory Generation

This chapter summarizes some of the recently reported techniques for the analysis and design of nonlinear dynamic systems. In particular, the differential geometric approaches to the analysis of nonlinear dynamic systems have been reviewed. The stress is on the concept of feedback linearization, which provides for a way to "linearize" nonlinear dynamic systems. In this context, *differential* and *orbital flatness* are briefly described. These concepts are a direct extension of the general feedback linearization approach, and form the basis for the optimization strategy proposed in the following chapter. The chapter also introduces the reader to a number of algorithms that may be used to characterize the *flatness* of a given system. These algorithms, though providing different approaches to the characterization problem, yield the so-called *flat outputs* for a dynamic system (if they exist). Among these approaches, the ones proposed by Guay *et al.* [1997] and Guay [1999] provide for a systematic look at the *flatness* characterization problem, and are applicable to a wide variety of systems.

Differential flatness is a very desirable property, since the knowledge of the linearizing (or *flat*) outputs provides an algebraic parameterization of the dynamics of the system. In other words, *flatness* of a system enables an algebraic mapping of any trajectory in the *flat output* space to the corresponding system variable trajectories. The need for explicit integration of the system differential equations is thus eliminated. Many researchers have identified the usefulness of this concept, and have applied it to answer some questions posed by the trajectory tracking problems. Fliess *et al.* [1993] have given a number of

applications of *differential flatness* in the control of trailer systems. The approach has also been considered by Rothfuss *et al.* [1996], where trajectories for *differentially flat* chemical reactors were assigned. Using *flatness*, van Nieuwstadt and Murray [1996] have considered the problem of assigning trajectories in real-time for various mechanical systems. Some of the more recent work has investigated the application of *flatness* to optimal trajectory generation for real-time applications [Agrawal *et al.*, 1999]. The approach, however, focused only on linear controllable time-invariant systems. In the area of *orbital flatness*, recent work presented by Murray [1999] considered the use of time scale transformations to handle the problem of rate and magnitude saturations in trajectory generation and tracking problems.

This thesis provides an approach to solve the optimal trajectory generation problem for the special class of *flat* nonlinear dynamic processes. The approach is an easier and more widely applicable extension to the one proposed by Agrawal *et al.* [1999]. The following chapters provide the more detailed explanations, as well as the description of the algorithm as applied to off-line and on-line trajectory optimization problems.

Chapter 3

Trajectory Optimization

"I have yet to see any problem, however complicated, which, when you looked at in the right way, did not become still more complicated." — Poul Anderson

The determination of optimal setpoint trajectories is an important control problem for many chemical processes [Terwiesch *et al.*, 1994] (*e.g.*, process start-up or shutdown, batch process control, *etc.*). The importance of these problems in chemical engineering can be gauged by the numerous research publications which deal with the solution to such problems. Rippin [1983] and Soroush and Kravaris [1993] provide exhaustive surveys on batch optimization with a perfect model and Terwiesch *et al.* [1994] gives a review of batch optimization with imperfect modelling. Nowadays, many of these chemical engineering process control applications have very tight specifications, due to market demands and/or environmental regulations, that cannot be satisfied by linear control methods [Fikar *et al.*, 1998]. One of the reasons for the failure of linear control in these applications is that the models of these processes are usually described by systems of highly nonlinear dynamic differential and algebraic equations in the state and manipulated variables. Additionally, complexities in the form of constraints on both the control and state profiles are also often present. This has led to increased activity in the development of nonlinear optimal control methods.

Among many methods, nonlinear predictive control has been particularly successful for some control applications [Michalska and Mayne, 1993]. The determination of optimal control trajectories for batch processes, however, are somewhat beyond the scope of standard predictive control schemes, mainly because of the large computation load [Fikar *et al.*, 1998]. This is mainly due to the differential-algebraic nature of the resulting optimization problems for which numerical techniques involving parameterization, discretization and/or forward integration of differential equations at every iteration, have to be employed. Chapter 1 provided a brief overview of some of these techniques used to solve Differential-Algebraic Optimization Problems (DAOP). This chapter builds up on Chapter 2 and introduces an alternative method to solve such DAOPs. Using concepts discussed in Chapter 2, the differential equations are eliminated from the optimization problem. The property of *flatness* is

exploited to transform the dynamic system into an equivalent dynamic system characterized by algebraic equations only. This transformed system presents an algebraic equivalent of the DAOP, which will be referred to as a *Normalized Dynamic Optimization* (NDO) problem in this thesis. The NDO problem is shown to facilitate optimal trajectory generation tasks for a class of nonlinear DAOPs.

The chapter is organized as follows. Starting from the statement of the DAOP, an algorithm is presented that exploits the *flatness* of the dynamics to transform the optimization problem into a problem formulation that is more easily solved. The following sections provide descriptions of the steps in the algorithm and illustrate them on specific examples. The step-wise application of this algorithm is illustrated on four DAOPs from mechanical and chemical engineering literature. The section also makes a comparison between the solutions obtained by this technique to the ones reported by the application of other techniques. The chapter ends with a discussion of the computational issues associated with the proposed algorithm, and some areas of further research. The chapter, by way of the algorithm, gives a basis for the real-time optimal trajectory generation problem which is the subject of next chapter.

3.1 Trajectory Optimization Problem

In batch processes, transient behaviour dominates and usually no steady-state is reached. Hence, the objective function needs to be optimized with respect to a dynamic model. The objective in the optimization of batch processes is the determination of input trajectories, that lead to the optimization of some objective function (*e.g.*, minimization of cost, maximization of product yield, *etc.*). This problem, as discussed in Chapter 1 can be represented as the following general differential-algebraic optimization problem (DAOP):

$$
\begin{array}{lll}
\underset{\mathbf{x}(t),\mathbf{u}(t)}{min} & \Phi(\mathbf{x}(t),\mathbf{u}(t)) & t \in [t_0, t_f] \\
s.t.: & & \\
& \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t)) & \mathbf{x}(t_0) = \mathbf{x}_0 \\
& \mathbf{g}(\mathbf{x}(t),\mathbf{u}(t)) \leq 0 & \\
& \mathbf{c}(\mathbf{x}(t),\mathbf{u}(t)) = 0 & \\
& \mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U & \\
& \mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U &
\end{array}
\tag{3.1}
$$

where, Φ represents the objective function, $\mathbf{g}$ and $\mathbf{c}$ are vectors containing functions in the algebraic constraints, $\mathbf{x}$ are the process states and $\mathbf{u}$ are the process inputs. The subscripts L and U indicate lower and upper bounds, respectively.

The DAOPs such as the one in Problem (3.1) are very difficult to solve with conventional analytical or numerical techniques. Moreover, these problems cannot be solved directly by typical nonlinear programming (NLP) techniques or optimal control methods [Cuthrell and Biegler, 1989]. Even when the profiles are discretized and treated as decision variables,

repeated, and often expensive, solution of the DAE model (with a large number of sensitivity equations) is still required. Optimal control methods, on the other hand, deal with continuous control profiles but normally cannot handle general algebraic constraints, such as the ones represented by the functions **c** or **g** [Cuthrell and Biegler, 1989]. Thus, special numerical techniques have to be employed in order to solve these problems. An overview of some of these methods with their advantages/disadvantages was presented in Chapter 1. In general, for large DAOPs (with many states and inputs), most of these methods can result in problem formulations that are computationally prohibitive (because of the large number of decision variables). Consequently, some form of decomposition or exploitation of problem structure is required to solve them efficiently. This problem becomes even more acute for real-time applications. In this thesis, an alternative algorithm is introduced which can solve a class of these optimization problems efficiently. In the proposed method, the concept of *flatness* (see Chapter 2) is used to express the system dynamics algebraically. In other words, the differential equations are eliminated from the optimization problem, hence making the problem completely algebraic.

3.2 Normalized Form Optimization

In this section, the *Normalized Dynamic Optimization* (NDO) algorithm is presented. This algorithm transforms the dynamic optimization problem to an algebraically constrained problem, which is more readily solved. The algorithm is enumerated in the following steps.

Algorithm: *Normalized Dynamic Optimization (NDO)*

1. **Identification of Flat Outputs:** The *flat outputs* ($\mathbf{y}(t)$) for the differential equation system, $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ are identified using the algorithm of Guay *et al.* [1997] or Rathinam and Murray [1999] . In case the system is not *differentially flat* in the original time scale t, the *orbital flatness* of the system is checked. For *orbital flatness*, a suitable time scaling is determined from the solution to equation: $\frac{dt}{d\tau} \equiv s(\mathbf{x}(t), \mathbf{u}(t)), \tau|_{t=0} \equiv \tau_0$ [Sampei and Furuta, 1986] and, if possible, the *flat outputs* ($\mathbf{y}(\tau)$) for the transformed system are identified[1].

2. **System Variable Transformation:** The original system variables $\mathbf{x}(t)$ and $\mathbf{u}(t)$ are written as functions of these *flat outputs* ($\mathbf{y}(t)$ or $\mathbf{y}(\tau)$) and a finite number of their derivatives:

$$\begin{aligned} \mathbf{x}(t) &= \boldsymbol{\alpha}(\mathbf{y}(t), \mathbf{y}^{(1)}(t), ...\mathbf{y}^{(k)}(t)) \equiv \boldsymbol{\alpha}(\bar{\mathbf{y}}(t)) \\ \mathbf{u}(t) &= \boldsymbol{\beta}(\mathbf{y}(t), \mathbf{y}^{(1)}(t), ...\mathbf{y}^{(k)}(t)) \equiv \boldsymbol{\beta}(\bar{\mathbf{y}}(t)) \end{aligned} \tag{3.2}$$

[1] *Differential* or *orbital flatness* of a system cannot be guaranteed. In fact, *differentially flat* systems form a small subset of all nonlinear systems. Time scaling merely expands this subset by accomodating a larger set of nonlinear systems in the *flatness* framework. See Chapter 2 for further details.

for the case of *differentially flat* systems. Here $\mathbf{y}^{(i)}(t)$ stands for the i^{th} derivative of $\mathbf{y}(t)$ with respect to t and k is the number of derivatives of $\mathbf{y}(t)$ required to represent the system in the form given in Equation (3.2). The *flat output* $\mathbf{y}(t)$ and its derivatives up to order k have been collected in the vector $\bar{\mathbf{y}}(t)$. For the *orbital flatness* case, which incorporates time scaling, the equivalent expressions take the form:

$$\begin{aligned} \mathbf{x}(t) &= \boldsymbol{\alpha}(\mathbf{y}(\tau), \mathbf{y}^{(1)}(\tau), \ldots \mathbf{y}^{(k)}(\tau)) \equiv \boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau)) \\ \mathbf{u}(t) &= \boldsymbol{\alpha}(\mathbf{y}(\tau), \mathbf{y}^{(1)}(\tau), \ldots \mathbf{y}^{(k)}(\tau)) \equiv \boldsymbol{\beta}(\bar{\mathbf{y}}(\tau)) \end{aligned} \tag{3.3}$$

where, the terminology is the same as in Equation (3.2), except that $\bar{\mathbf{y}}$ is now a function of $\tau, \bar{\mathbf{y}}(\tau)$ instead of t. In this case, the new time scale τ is obtained as a function of t from an analytical solution of the differential equation:

$$\frac{dt}{d\tau} = s(\mathbf{x}(t), \mathbf{u}(t)), \quad \tau|_{t=0} = \tau_0 \tag{3.4}$$

For the case when an analytical solution is not possible because of the complex nature of the problem (*e.g.*, complex time scaling function or the nature of the parameterization of the *flat outputs*, *etc.*), the equation (3.4) is solved numerically[2].

3. **Problem Transformation:** The functions $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are substituted for $\mathbf{x}(t)$ and $\mathbf{u}(t)$ in the original optimization problem. The differential equations in this model are already accounted for in the calculation of *flat outputs*, and hence are eliminated in this new form. The *normalized form* of the optimization problem can then can be represented as:

$$\begin{aligned} \min_{\bar{\mathbf{y}}} \quad & \Phi(\boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau)), \boldsymbol{\beta}(\bar{\mathbf{y}}(\tau))) \qquad \tau \in [\tau_0, \tau_f] \\ s.t.: \quad & \\ & \boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau_0)) = \mathbf{x}_0 \\ & \mathbf{g}(\boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau)), \boldsymbol{\beta}(\bar{\mathbf{y}}(\tau))) \leq 0 \\ & \mathbf{c}(\boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau)), \boldsymbol{\beta}(\bar{\mathbf{y}}(\tau))) = 0 \\ & \mathbf{x}_L \leq \boldsymbol{\alpha}(\bar{\mathbf{y}}(\tau)) \leq \mathbf{x}_U \\ & \mathbf{u}_L \leq \boldsymbol{\beta}(\bar{\mathbf{y}}(\tau)) \leq \mathbf{u}_U \end{aligned} \tag{3.5}$$

Note that the Problem (3.5) contains only algebraic constraints, yet is equivalent to Problem (3.1). Problem (3.5) will be referred to as the *Normalized Dynamic Optimization (NDO)* in the following discussions.

4. **Parameterization:** The *flat outputs* are parameterized by suitable functions of time (*e.g.*, simple polynomials, Lagrange polynomials, *etc.*). In this thesis, simple polynomials (*i.e.*, $\mathbf{y}(\tau) = \mathbf{a} + \mathbf{b}\tau + \mathbf{c}\tau^2 + \ldots$) are used to represent the *flat outputs*. Given an

[2]The differential flatness case (*flat outputs*, $\mathbf{y}(t)$) can be considered as just a special case of *orbital flatness* (*flat outputs*, $\mathbf{y}(\tau)$) by setting $\tau = t$. Hence, in the following steps and the discussion to follow, all the expressions and formulations are given in terms of $\mathbf{y}(\tau)$ with the assumption that for *differentially flat* systems, $\tau = t$ and for *orbitally flat* systems, τ as a function of t is given by the time scaling equation (3.4).

assumed structure for the *flat outputs*, the optimization problem is reduced to values for the parameter vector $\boldsymbol{\theta} = [\mathbf{a}^T, \mathbf{b}^T, \mathbf{c}^T, \ldots]^T$. Thus, Problem (3.5) becomes[3]:

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \Psi(\boldsymbol{\theta}, \tau) \qquad\qquad \tau \in [\tau_0, \tau_f] \\ s.t.: \quad & \\ & \mathbf{f}_{\mathbf{x}}(\tau_0) = \mathbf{x}_0 \\ & \mathbf{f}_{\mathbf{g}}(\boldsymbol{\theta}, \tau) \leq 0 \\ & \mathbf{f}_{\mathbf{c}}(\boldsymbol{\theta}, \tau) = 0 \\ & \mathbf{x}_L \leq \mathbf{f}_{\mathbf{x}}(\boldsymbol{\theta}, \tau) \leq \mathbf{x}_U \\ & \mathbf{u}_L \leq \mathbf{f}_{\mathbf{u}}(\boldsymbol{\theta}, \tau) \leq \mathbf{u}_U \end{aligned} \tag{3.6}$$

The Problem (3.6) is an equivalent form of Problem (3.1), yet it is only algebraically constrained. The parameter set, $\boldsymbol{\theta}$ forms the decision variable vector for this transformed optimization problem[4]. The choice of the optimization technique used for solving Problem (3.6) depends on the specific type of inequality constraints that are present in the problem. If the inequality constraints represent restrictions on the path that trajectories may follow within the optimization interval, *i.e.*, path constraints, the problem is a one-parametric Semi-Infinite Optimization problem [Jongen and Stein, 1997] , which is a sub-class of Semi-Infinite Optimization [Polak, 1997; Hettich and Kortanek, 1993] problems that are relatively easy to solve. On the other hand, when there are no path constraints in Problem (3.6), it is a conventional algebraic optimization problem that may be solved using standard LP, QP or NLP codes, as appropriate.

5. **Optimal Trajectory Determination:** Determination of the optimal trajectories from Problem (3.6) is a two step procedure. In the first step, the NDO Problem (3.6) must be solved numerically. The second step consists of substituting the calculated values of the parameter vector $\boldsymbol{\theta}$ into the expression for the *flat outputs* $\mathbf{y}$, which are in turn substituted into the expressions given in Equations (3.2) or (3.3) to determine the trajectories for the state and input variables.

3.3 Steps in the NDO Algorithm

This section gives a more detailed explanation of the NDO algorithm steps discussed in the previous section. The section is divided into three sections. The first section provides references to the relevant algorithms which have been used to characterize the *flatness* of the example systems. Also, it combines Steps 1 and 2 of the algorithm and illustrates the

[3]The functions are not explicitly stated here, but can be easily calculated. Here, the functional dependency of the variables on $\boldsymbol{\theta}, \tau$ will be shown with the relevant subscript to $\mathbf{f}$. For example, the constraint $\mathbf{g}$ in the original $\{\mathbf{x}, \mathbf{u}\}$ space will be represented as $\mathbf{f}_{\mathbf{g}}$ in the $\{\mathbf{y}\}$ space and similarly for others.

[4]The choice of the order of polynomial is dictated by the problem constraints and objective function value. The polynomial order, in general, has to be such that there is at least one degree of freedom for optimization. The parameter set can be further inflated to achieve better results from the algorithm.

application of *flat output* identification algorithms on both classes of systems, *viz.*, *differentially* and *orbitally flat*. The next section illustrates, via an example, the *normalization* of DAOP to an NDO problem. The next section discusses the method for the solution to hence obtained NDO to determine the optimal trajectories for the system.

3.3.1 Flatness and System Transformation

This section discusses the first two steps of the NDO algorithm: 1) the identification of *flat outputs* for the system; and using these *flat outputs*, the 2) system transformation to the *flat output* space. In the following discussion, the identification of the *differentially flat* and *orbitally flat outputs* via these algorithms is illustrated on examples. The next step of representing the system variables as functions of these *flat outputs*[5] and a finite number of their derivatives, is also illustrated for both *differential* and *orbital flatness* cases.

Example 3.3.1 - Differentially Flat System

Consider the system of Charlet *et al.* [1989; 1991] , which was previously discussed in Chapter 2:

$$\begin{aligned} \dot{x}_1 &= x_2 + x_3 u_2 \\ \dot{x}_2 &= x_3 + x_1 u_2 \\ \dot{x}_3 &= u_1 + x_2 u_2 \\ \dot{x}_4 &= u_2. \end{aligned} \tag{3.7}$$

The *flat outputs* for this system were found to be: $y_1 = x_4$ and $y_2 = x_2 u_2 - x_1$ in Chapter 2. Here, the steps of the Guay *et al.* [1997] algorithm are illustrated on this example to obtain the same *flat outputs*. The steps have been taken directly from a previous publication [Guay, 1996] with minor modifications, and are presented here just for illustration purposes. The following symbolic computations have been performed in $Maple^{TM}$ using the "*liesymm*" package. The Pfaffian system (see Appendix A for details) associated with the system given in Equation (3.7) is given by:

$$I \equiv \begin{bmatrix} d(x_1) - (x_2 + x_3 u_2) d(t) \\ d(x_2) - (x_3 + x_1 u_2) d(t) \\ d(x_3) - (u_1 + x_2 u_2) d(t) \\ d(x_4) - u_2 d(t) \end{bmatrix} \equiv \begin{bmatrix} w1 \\ w2 \\ w3 \\ w4 \end{bmatrix} \tag{3.8}$$

where $d(.)$ stands for the exterior derivative. In the following steps, $\wedge$ stands for the wedge product and *mod* for *modulo*. An introduction to the language of exterior calculus is given in Appendix A, which should be referred for details about any of these. Also, Bryant *et al.* [1991] provides a comprehensive source of information for exterior differential systems. From the definitions of exterior derivatives, wedge products and *modulo*, the following condition

[5] In this thesis, *flat* systems will be assumed to encompass both *differentially flat* and *orbitally flat* systems. From hereon, *flat* will imply *differentially flat* or *orbitally flat*, depending on the context.

is obtained:

$$\begin{bmatrix} d(w_1) \\ d(w_2) \\ d(w_3) \\ d(w_4) \end{bmatrix} = d(t) \wedge \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \end{bmatrix} d(u_1) mod\ I, d(u_2) \tag{3.9}$$

From the above, the first derived flag for this system, $I^{(1)} \equiv [w_1, w_2, w_4]$ is hence constructed. Proceeding with the derived flag, the following is obtained:

$$\begin{bmatrix} d(w_1) \\ d(w_2) \\ d(w_4) \end{bmatrix} \equiv d(t) \wedge \begin{bmatrix} u_2 \\ 1 \\ 0 \end{bmatrix} d(x_3) mod\ I^{(1)}, d(u_2) \tag{3.10}$$

To choose the next derived flag, $I^{(2)}$ for the system, a new generator, w_5 is defined as a linear combination of the first two generators, w_1 and w_2 as: $w_5 = w_1/u_2 - w_2$. With this definition, the following condition is obtained:

$$d(w_5) = 0 mod\ I^{(1)}, d(u_2) \tag{3.11}$$

that permits the definition of the second derived system as: $I^{(2)} = [w_4, w_5]$. The following condition is then obtained:

$$\begin{bmatrix} d(w_4) \\ d(w_5) \end{bmatrix} \equiv d(t) \wedge \begin{bmatrix} 0 \\ \frac{1}{u_2^2} - u_2 \end{bmatrix} d(x_1) mod\ I^{(2)}, d(u_2) \tag{3.12}$$

which implies that the bottom derived system, $I^{(2)}$ is integrable. Hence, the system fulfils the conditions given by Guay *et al.* [1997] and is therefore dynamic feedback linearizable (or *differentially flat*). The generators of the bottom derived system, $I^{(2)}$ are given by:

$$\begin{bmatrix} w_4 \\ w_5 \end{bmatrix} \equiv \begin{bmatrix} d(x_4) - u_2 d(t) \\ \frac{d(x_1)}{u_2} - d(x_2) + (-\frac{x_2}{u_2} + x_1 u_2) d(t) \end{bmatrix} \tag{3.13}$$

implying that the linearizing output functions (*flat outputs*) for this system are:

$$\begin{aligned} \hat{y}_1 &\equiv x_4 \\ \hat{y}_2 &\equiv \frac{x_1}{u_2} - x_2 \end{aligned} \tag{3.14}$$

which can be converted to the ones reported in Chapter 2, with the following transformation:

$$\begin{aligned} y_1 &\equiv \hat{y}_1 = x_4 \\ y_2 &\equiv -\dot{y}_1 \hat{y}_2 = -u_2 \hat{y}_2 = x_2 u_2 - x_1 \end{aligned} \tag{3.15}$$

With this set of *flat outputs* $\{y_1, y_2\} \equiv \{x_4, x_2 u_2 - x_1\}$, as shown previously in chapter 2, the system variables can be written as functions of these *flat outputs* and a finite number

of their derivatives as:[6]

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} \dot{y}_1 \left(\dfrac{\ddot{y}_2 + \dot{y}_1^2 y_2}{\ddot{y}_1 + \dot{y}_1^3 - 1} \right) - y_2 \\ \dfrac{\ddot{y}_2 + \dot{y}_1^2 y_2}{\ddot{y}_1 + \dot{y}_1^3 - 1} \\ \dfrac{\dot{x}_1 - x_2}{u_2} = f_1(\bar{y}_1, \bar{y}_2) \\ y_1 \end{bmatrix}; \quad \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \dot{x}_3 - x_2 u_2 = f_2(\bar{y}_1, \bar{y}_2) \\ \dot{y}_1 \end{bmatrix} \tag{3.16}$$

The example illustrates how the algorithm of Guay *et al.* [1997] can be applied, systematically, to generate the *flat outputs* for a given dynamic system.

Example 3.3.2 - Orbitally Flat System

Consider the tubular chemical reactor problem that has been studied by several researchers [Ray, 1981; Biegler, 1984; Logsdon and Biegler, 1989;1992; Dadebo and McAuley, 1995] in which the following reactions take place:

$$A \xrightarrow{k_1} B,$$
$$A \xrightarrow{k_2} C.$$

Assuming that $x_1 \equiv C_A/C_{Af}$ and $x_2 \equiv C_B/C_{Af}$, the system dynamics can be represented as:

$$\begin{aligned} \frac{dx_1}{dt} &= -ux_1 - \frac{1}{2}u^2 x_1 \\ \frac{dx_2}{dt} &= ux_1 \end{aligned} \tag{3.17}$$

where the control $u = k_1 L/v$ (L is the reactor length and v is the space velocity within the reactor). This system is not *differentially flat*, but can be shown to be *orbitally flat* with the time scaling:

$$\frac{dt}{d\tau} = s(\mathbf{x}, \mathbf{u}) = \frac{1}{u^2}, \ \tau|_{t=0} = 0 \tag{3.18}$$

The system, in the new time scale τ can be represented as:

$$\begin{aligned} \frac{dx_1}{d\tau} &= -\frac{x_1}{u} - \frac{1}{2}x_1 \\ \frac{dx_2}{d\tau} &= \frac{x_1}{u} \end{aligned} \tag{3.19}$$

Now, using the transformed dynamics in Equation (3.19), the Pfaffian system is given by:

$$I \equiv \begin{bmatrix} d(x_1) - (-\dfrac{x_1}{u} - \dfrac{1}{2}x_1)d(\tau) \\ d(x_2) - (\dfrac{x_1}{u})d(\tau) \end{bmatrix} \equiv \begin{bmatrix} w1 \\ w2 \end{bmatrix} \tag{3.20}$$

[6] In the following expressions, only the functional dependence is shown because the resulting expressions are quite large. Further, the outputs y_1, y_2 and their derivatives upto the second order have been stacked in $\bar{y}_1$ and $\bar{y}_2$ respectively, i.e. $\bar{y}_1 = \{y_1, \dot{y}_1, \ddot{y}_1\}$ and $\bar{y}_2 = \{y_2, \dot{y}_2, \ddot{y}_2\}$.

With this definition of the Pfaffian, the following condition is obtained:

$$\begin{bmatrix} d(w_1) \\ d(w_2) \end{bmatrix} = d(\tau) \wedge \begin{bmatrix} -\frac{x_1}{u^2} \\ \frac{x_1}{u} \end{bmatrix} d(u) mod\ I \tag{3.21}$$

To define the first derived flag for this system, a new generator that is a linear combination of w_1 and w_2 is defined: $w_3 = w_1 + w_2$. With this definition and the condition that $d(w_3) = 0 mod\ I$, the first derived flag for this system, $I^{(1)} \equiv [w_3]$ is hence constructed. Proceeding with the derived flag, the following is obtained:

$$\begin{bmatrix} d(w_3) \end{bmatrix} \equiv d(\tau) \wedge \begin{bmatrix} -\frac{1}{2} \end{bmatrix} d(x_2) mod\ I \tag{3.22}$$

which implies that the bottom derived system, $I^{(1)}$ is integrable. Hence, the system fulfils the conditions given by Guay [1999] and is therefore *orbitally flat*. The generator of the bottom derived system, $I^{(1)}$ is given by:

$$w_3 \equiv d(x_1) + d(x_2) - \left[\frac{(-ux1 - \frac{1}{2}u^2x1)}{u^2} + \frac{x_1}{u} \right] d(\tau) \tag{3.23}$$

implying that the linearizing output function (*flat output*) for this system is:

$$y = x_1 + x_2 \tag{3.24}$$

The fact that the output (3.24) does form a *flat output* for the system (3.19) is illustrated by writing the system variables as functions of this *flat output* and a finite number of its derivatives. As a first step, the *flat output* y is differentiated with respect to τ (it must be noted that y is a function of τ which in turn is related to t by Equation (3.18)) to obtain:

$$\frac{dy}{d\tau} = \frac{dx_1}{d\tau} + \frac{dx_2}{d\tau} = -\frac{1}{2}x_1$$

which can be rearranged to obtain:

$$x_1 = -2\dot{y} \tag{3.25}$$

By substituting Equation (3.25) into Equation (3.24) and simplifying, the following expression for x_2 is obtained:

$$x_2 = y + 2\dot{y} \tag{3.26}$$

Equations (3.25) or (3.26) can then be substituted into either expression in Equation (3.19) and simplified to yield an expression for the input variable:

$$u = \frac{-2\dot{y}}{\dot{y} + 2\ddot{y}} \tag{3.27}$$

Hence, $y = x_1 + x_2$ does form a *flat output* for the system. The system variables as functions of the *flat output* and its derivatives are given by Equations (3.25), (3.26) and (3.27). The new time scale τ is obtained as a solution to the differential Equation (3.18):

$$\frac{dt}{d\tau} = \frac{1}{u^2(t)} = \left(\frac{\dot{y}(\tau) + 2\ddot{y}(\tau)}{-2\dot{y}(\tau)}\right)^2 \tag{3.28}$$

It is thus clear that the algorithms of Guay *et al.* [1997] and Guay [1999] can be applied systematically to characterize the flatness of dynamic systems as well as identify the *flat outputs* for them. The two examples are presented here for illustration purposes without giving many other details of the algorithms. The reader should consult the original publications for further reference.

3.3.2 Normalization of the DAOP Problem

Normalization consists of mapping the optimization problem for the original system (in terms of original system variables $\mathbf{x}$ and $\mathbf{u}$) to one in the *flat output* ($\mathbf{y}(\tau)$) space. Putting it simply, it refers to the process of substituting the expressions in Equation (3.3) in the DAOP (3.1). The technique is illustrated on the tubular reactor example in the following section.

Example 3.3.3 - Parallel Reaction Problem

The dynamic optimization problem for the *orbital flatness* example (*cf.*, Example 3.3.2) is:

$$\begin{aligned}
&\underset{u(t)}{max} \quad x_2(t_f); \qquad t_f = 1; \qquad t \in [0,1] \\
&s.t.: \\
&\qquad \dot{x}_1(t) = -u(t)x_1(t)\left[1 + \frac{1}{2}u(t)\right] \\
&\qquad \dot{x}_2(t) = u(t)x_1(t) \\
&\qquad 0 \leq u(t) \leq 5 \\
&\qquad x_1(0) = 1 \\
&\qquad x_2(0) = 0
\end{aligned} \tag{3.29}$$

It was shown in Example 3.3.2 that the *flat output* for this system is given by Equation (3.24). It was also shown in Equations (3.25), (3.26) and (3.27) that the system variables can be written as functions of this *flat output* and its derivatives. Substitution of the expressions for x_1, x_2 and u into Problem (3.29) and simplifying, yields the NDO problem:

$$\begin{aligned}
&\underset{y(\tau),\dot{y}(\tau),\ddot{y}(\tau)}{max} \quad y(\tau_f) + 2\dot{y}(\tau_f) \quad \tau \in [0,\tau_f] \\
&s.t.: \\
&\qquad \ddot{y}(\tau) \geq \frac{-7}{10}\dot{y}(\tau) \\
&\qquad \dot{y}(\tau) \leq 0 \\
&\qquad y(0) = 1
\end{aligned} \tag{3.30}$$

where τ is obtained as a function of t from the solution of Equation (3.28) and final time τ_f corresponds to $t = 1$. The next section discusses how this problem can be solved by a suitable parameterization of the *flat output*.

3.3.3 Parameterization and Trajectory Determination

In order to solve the NDO Problem (3.5), a suitable parameterization of the *flat outputs* is carried out. As discussed in the algorithm, the parameterization can take one of the many forms without affecting the basic solution scheme. In this thesis, the *flat outputs* are parameterized in time as: $\mathbf{y}(\tau) = \mathbf{a} + \mathbf{b}\tau + \mathbf{c}\tau^2 + \dots$, with the parameter vector denoted by $\boldsymbol{\theta} = [\mathbf{a}^T, \mathbf{b}^T, \mathbf{c}^T, \dots]^T$. With this type of parameterization, the NDO Problem (3.5) is equivalent to the form in Equation (3.6). This problem can then be solved using available nonlinear (or linear) Semi-Infinite Optimization Codes. This step is illustrated on the tubular reactor example (*cf.*, Example 3.3.2) in the following example.

Example 3.3.4 - Parallel Reaction Problem

Consider the NDO problem formulation of Equation (3.30). For illustration purposes, suppose $y(\tau)$ in (3.30) can be represented in quadratic form:

$$y(\tau) = a + b\tau + c\tau^2 \tag{3.31}$$

Then, the expressions for its derivatives up to the second order are given by:

$$\begin{aligned} \dot{y}(\tau) &= b + 2c\tau \\ \ddot{y}(\tau) &= 2c \end{aligned} \tag{3.32}$$

and the time scaling can be determined via the analytical solution of Equation (3.28) to be:

$$t = -\frac{\tau}{4} + \frac{2c}{b + 2c\tau} - \ln(b + 2c\tau) \tag{3.33}$$

Substituting Equations (3.31), (3.32) and (3.33) into the NDO Problem (3.30) yields:

$$\begin{aligned} \underset{a,b,c}{max} \quad & a + (2 + \tau_f)b + (4\tau_f + \tau_f^2)c \\ s.t.: \quad & \\ & a = 0 \\ & \frac{7}{10}b + 2(1 + \frac{7}{10}\tau)c \geq 0 \\ & b + 2c\tau \leq 0 \\ & \frac{2c}{b} - \ln(b) = 0 \\ & -\frac{\tau_f}{4} + \frac{2c}{b + 2c\tau_f} - \ln(b + 2c\tau_f) = 1 \end{aligned} \tag{3.34}$$

where the final two equations in Problem (3.34) represent the initial and final conditions on the solution to the time scaling problem given in Equation (3.18). Problem (3.34) is an NLP problem in two decision variables, as the parameter a is required to be zero, and

can be solved using available nonlinear Semi-Infinite Optimization codes. If the system had been *differentially flat*, then Problem (3.34) would reduce to a simple linear Semi-Infinite Optimization problem.

The solution of Problem (3.34) required solving a differential equation for the time scaling, which was a result of the systems orbital *flatness*. *Differentially flat* systems will not require the solution of any such differential equations. In this example, the differential equation could be solved analytically; however this will not be the general case. When an analytical solution is not possible, a numerical differential equation solver must be embedded within the optimization problem formulation. The additional computation load that results from embedding a numerical differential equation solver within the problem formulation is comparatively small however, as the time scaling differential equation is scalar.

An appropriate Semi-Infinite Optimization Technique, as discussed above, yields the optimal value of the parameter vector $\boldsymbol{\theta}$. This solution thus defines the *flat output* trajectory $(y(\tau))$ as well as the trajectories for its derivatives $(\dot{y}(\tau), \ddot{y}(\tau), \dots)$ from the parameterization information. The optimal input and state trajectories are hence obtained uniquely from Equation (3.2) or (3.3). This step will be illustrated in the next section, where optimal trajectories for a number of benchmark case studies taken from literature would be obtained using this technique.

3.4 Illustrative Examples

In this section, the proposed NDO scheme is illustrated on a number of benchmark case studies from the dynamic optimization literature. These examples are intended to provide a comparison of the proposed NDO method to the more established approaches. The comparison will be made in terms of accuracy of the solution and the size of the optimization problem to be solved.

3.4.1 Single Integrator System

Consider the single integrator system studied by Goh and Teo [1988], Luus [1991] and Dadebo and McAuley [1995]. The optimization problem for the system can be written as:

$$\begin{aligned} \min_{u(t)} \quad & x_2(t_f); \qquad t_f = 1 \\ s.t.: \quad & \\ & \dot{x}_1(t) = u(t) \\ & \dot{x}_2(t) = x_1^2(t) + u^2(t) \\ & x_1(0) = 1 \\ & x_2(0) = 0 \\ & x_1(1) = 1 \end{aligned} \tag{3.35}$$

This system can be thought of as having a *flat output:* $y(t) \equiv x_1(t)$, since the initial condition for $x_2(t)$ is known. Thus, an expression for $x_2(t)$ can be obtained as an integral

of a function of $y(t) \equiv x_1(t)$ and its first derivative. The system variables can then be expressed as functions of the *flat output* $y(t)$ as follows:

$$x_1(t) \equiv y(t) \tag{3.36}$$

$$x_2(t) = \int \left[y^2(t) + \dot{y}^2(t)\right] dt \tag{3.37}$$

$$u(t) = \dot{y}(t) \tag{3.38}$$

Parameterizing $y(t)$ as:

$$y(t) \equiv a + bt + ct^2 + dt^3 + et^4 \tag{3.39}$$

and collecting the coefficients of the parameterization in a vector: $\boldsymbol{\theta} \equiv [a, b, c, d, e]^T$ enables the dynamic optimization Problem (3.35) to be written as:

$$\begin{array}{ll} \underset{\boldsymbol{\theta}}{min} & \boldsymbol{\theta}^T \mathbf{H} \boldsymbol{\theta} \\ s.t.: & \\ & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \boldsymbol{\theta} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{array} \tag{3.40}$$

where:

$$\mathbf{H} = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} \\ \frac{1}{4} & \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} \\ \frac{1}{5} & \frac{1}{6} & \frac{1}{7} & \frac{1}{8} & \frac{1}{9} \end{bmatrix}$$

The problem in Equation (3.40) does not contain any path constraints and is an equality constrained QP problem, which can easily be solved either analytically or using a number of readily available optimization codes. The quadratic programming solution to the Problem (3.40) is $\boldsymbol{\theta} \equiv [1, -0.4621, 0.4996, -0.0749, 0.0375]^T$, which yields an objective function value of $x_2(1) = 0.9242343146$. The analytical solution to this problem is found to be identical to ten significant figures. Figure 3.1 gives a comparison between the state and input trajectories determined using the optimal parameter values in Equations (3.36) through (3.38) and the trajectories determined analytically. The analytical solution can be found by reformulating Problem (3.35) as a finite time optimal control problem. Problem (3.35) is equivalent to:

$$\begin{array}{ll} \underset{u(t)}{min} & \int_0^1 \left[x_1^2(t) + u^2(t)\right] dt \\ s.t.: & \\ & \dot{x}_1(t) = u(t) \\ & x_1(0) = 1 \\ & x_1(1) = 1 \end{array} \tag{3.41}$$

The problem in Equation (3.41) can be solved via least squares optimal control theory [Brockett, 1969], which involves the solution of a Riccati differential equation. The proposed

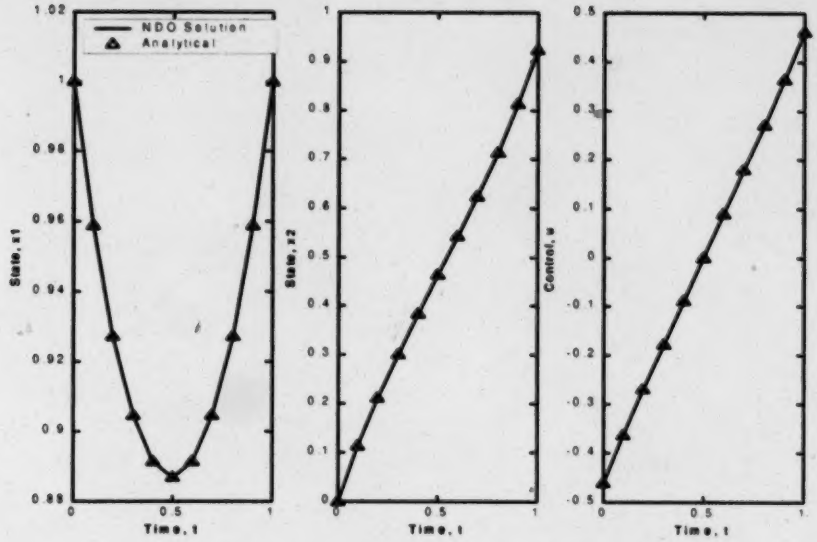


Figure 3.1: Comparison of Trajectories for the Single Integrator System.

method converted an optimal control problem into an equality constrained QP problem in only five decision variables (*i.e.*, the coefficients in the polynomial representation for the *flat output*), which yielded very high solution accuracy despite the low order approximation. This example may thus have some interesting implications for the solution of linear optimal control problems.

3.4.2 Parallel Reaction Problem

This problem, with bounds on the control variable, has been studied by several researchers including Ray [1981], Biegler [1984], Logsdon and Biegler [1989;1992] and Dadebo and McAuley [1995]. In this example, a tubular reactor makes two products according to the parallel reaction scheme: $A \xrightarrow{k_1} B, A \xrightarrow{k_2} C$. The optimization problem for this system can be represented as:

$$\begin{aligned} &\underset{u(t)}{max} \quad x_2(t_f); \qquad t_f = 1; \qquad t \in [0,1] \\ &s.t.: \\ &\qquad \dot{x}_1(t) = -u(t)x_1(t)\left[1 + \frac{1}{2}u(t)\right] \\ &\qquad \dot{x}_2(t) = u(t)x_1(t) \\ &\qquad 0 \leq u(t) \leq 5 \\ &\qquad x_1(0) = 1 \\ &\qquad x_2(0) = 0 \end{aligned} \tag{3.42}$$

where the manipulated variable is defined as $u(t) \equiv k_1 L/v$ (L is the reactor length and v is the space velocity), the first state variable is defined as $x_1 \equiv C_A/C_{Af}$ and the second

state variable is defined as $x_2 \equiv C_B/C_{Af}$. One choice of linearizing output for this system is $y(t) \equiv x_1(t)$. The system variables can be expressed in terms of the linearizing output $y(t)$ and its derivatives as follows[7]. The first state ODE is stated in terms of $y(t)$ and $\dot{y}(t)$:

$$\frac{1}{2}u^2(t) + u(t) + \frac{y(t)}{\dot{y}(t)} = 0 \tag{3.43}$$

which specifies:

$$u(t) = -1 + \sqrt{1 - 2\frac{\dot{y}(t)}{y(t)}}. \tag{3.44}$$

Next, the substitution of Equation (3.44) into the ODE for the second state yields:

$$\dot{x}_2(t) = \left(-1 + \sqrt{1 - 2\frac{\dot{y}(t)}{y(t)}}\right) y(t) \tag{3.45}$$

Representing the linearizing output $y(t)$ in quadratic form:

$$y(t) \equiv 1 + at + bt^2 \tag{3.46}$$

and substituting in Equation (3.45) yields:

$$\dot{x}_2(t) = \left(-1 + \sqrt{1 - 2\frac{b + 2ct}{a + bt + ct^2}}\right)(a + bt + ct^2). \tag{3.47}$$

Integrating using the initial condition $x_2(0) = 0$ gives:

$$x_2(t) = \int_0^t \left(-1 + \sqrt{1 - 2\frac{b + 2c\sigma}{a + b\sigma + c\sigma^2}}\right)(a + b\sigma + c\sigma^2)d\sigma. \tag{3.48}$$

which can be solved by quadrature. Collecting the coefficients in a vector $\boldsymbol{\theta} \equiv [a, b, c]^T$ enables the dynamic optimization Problem (3.42) to be written as:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & \int_0^1 \left(-1 + \sqrt{1 - 2\tfrac{b+2c\sigma}{a+b\sigma+c\sigma^2}}\right)(a + b\sigma + c\sigma^2)d\sigma \\ s.t.: \quad & \\ & a = 1 \\ & 0 \leq \left(-1 + \sqrt{1 - 2\tfrac{b+2ct}{a+bt+ct^2}}\right) \leq 5 \end{aligned} \tag{3.49}$$

Problem (3.49) does contain input path constraints and is a Semi-Infinite Optimization problem. Using one of the readily available Semi-Infinite Optimization codes, the objective function value obtained for the optimal solution to this problem is $x_2(1) = 0.57189$ and the optimal polynomial coefficient values are $\boldsymbol{\theta} \equiv [1, -1.03180, 0.081453]^T$. The optimal state and input trajectories determined using this technique are shown in Figure 3.2.

[7] It must be noted that the system dynamics were identified as being *orbitally flat* in Example 3.3.2. In this example, however as shown, the system is *differentially flat* because of the given initial conditions for the differential equation constraints.

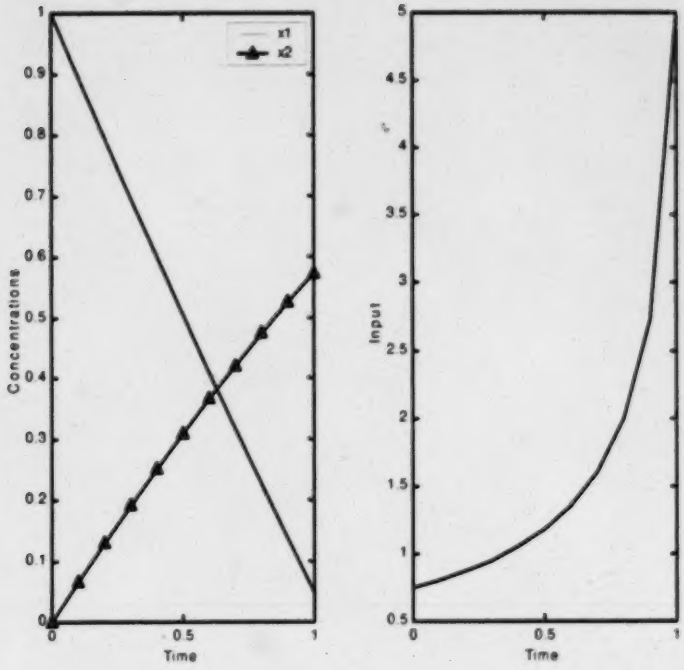


Figure 3.2: Optimal Trajectories for the Parallel Reaction Problem.

The simple, quadratic approximation produced accurate results with very few computations. Further the solution to the NDO Problem (3.49) required, only the integration of the objective function and was computationally inexpensive. Increasing the order of the polynomial used to represent the *flat output* to tenth order, produced an optimum solution with an objective function value of $x_2(1) = 0.57304$ and optimal polynomial coefficients of:

$$\begin{aligned}\boldsymbol{\theta} \equiv\ & [1, -1.06433, 0.15822, 0.022151, 0.15497, . - 0.097468, -0.16193, \\ & -0.093541, -0.19312, -0.10974, 0.41606]^T\end{aligned}$$

which is slightly less than the reported literature values.

3.4.3 Consecutive Reaction Problem

Consider the batch reactor problem given in Ray [1981], in which takes place the consecutive reaction:

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

The operating objective for this reactor is to obtain the reactor temperature progression that maximizes the intermediate product for a fixed batch time. The optimization problem

for the system can be stated as:

$$\begin{aligned} \max_{u(t)} \quad & x_2(t_f); \qquad t_f = 1 \\ s.t.: \quad & \\ & \dot{x}_1(t) = -u(t)x_1^2(t) \\ & \dot{x}_2(t) = -0.03875u^2(t)x_2(t) + u(t)x_1^2(t) \\ & x_1(0) = 1 \\ & x_2(0) = 0 \\ & 0.9092 \leq u(t) \leq 7.4831 \end{aligned} \tag{3.50}$$

where the manipulated variable is defined as $u \equiv 4000e^{\frac{-2500}{T}}$, and the state variables are defined as $x_1 \equiv c_A$ and $x_2 \equiv c_B$, respectively.

This system can be shown to be *orbitally flat* using the time scaling transformation:

$$\frac{dt}{d\tau} \equiv \frac{1}{u^2(t)}, \quad \tau|_{t=0} = 0 \tag{3.51}$$

A *flat output* for the transformed system be defined as $y(\tau) = x_1(t) + x_2(t)$. Then, representing the output $y(\tau)$ in the cubic form:

$$y(\tau) \equiv a + b\tau + c\tau^2 + d\tau^3, \tag{3.52}$$

allows the system variables to be expressed as:

$$\begin{aligned} x_1(t) &= y(\tau) + \frac{1}{0.03875}\dot{y}(\tau) \\ x_2(t) &= -\frac{1}{0.03875}\dot{y}(\tau) \\ u(t) &= \frac{\left(-y(\tau) - \frac{1}{0.03875}\dot{y}(\tau)\right)^2}{y(\tau) + \frac{1}{0.03875}\ddot{y}(\tau)}. \end{aligned} \tag{3.53}$$

Note that $b = 0$ ensures that $x_2(0) = 0$ and $a = 1$ ensures that $x_1(0) = 1$. Collecting the remaining coefficients in a vector $\boldsymbol{\theta} \equiv [c, d]^T$ enables the dynamic optimization Problem (3.50) to be written as:

$$\begin{aligned} \max_{\boldsymbol{\theta}} \quad & -\frac{1}{0.03875}\dot{y}(\tau_f) \\ s.t.: \quad & \\ & \frac{dt}{d\tau} \equiv \left[\frac{y(\tau) + \frac{1}{0.03875}\ddot{y}(\tau)}{\left(-y(\tau) - \frac{1}{0.03875}\dot{y}(\tau)\right)^2}\right]^2, \quad \tau|_{t=0} = 0 \\ & 0.9092 \leq \frac{\left(-y(\tau) - \frac{1}{0.03875}\dot{y}(\tau)\right)^2}{y(\tau) + \frac{1}{0.03875}\ddot{y}(\tau)} \leq 7.4831 \\ & 0 \leq t \leq 1 \end{aligned} \tag{3.54}$$

The transformed optimization problem requires the integration of one differential equation, which is needed for the synchronization of the two time scales. The computational advantage of the proposed approach is that it only requires the integration of a scalar ordinary differential equation, regardless of the underlying dimension of the optimization problem.

Problem (3.54) can be solved via Semi-infinite optimization; however the final time in the new time scale (τ_f) becomes an additional decision variable in the optimization problem. The optimal objective function value for the solution is $x_2(1) = 0.610203$ and optimal parameter values are $\boldsymbol{\theta} = [-0.12919, 0.03883]^T$ and $\tau_f = 6.00053$. The calculated system trajectories are shown in Figure 3.3.

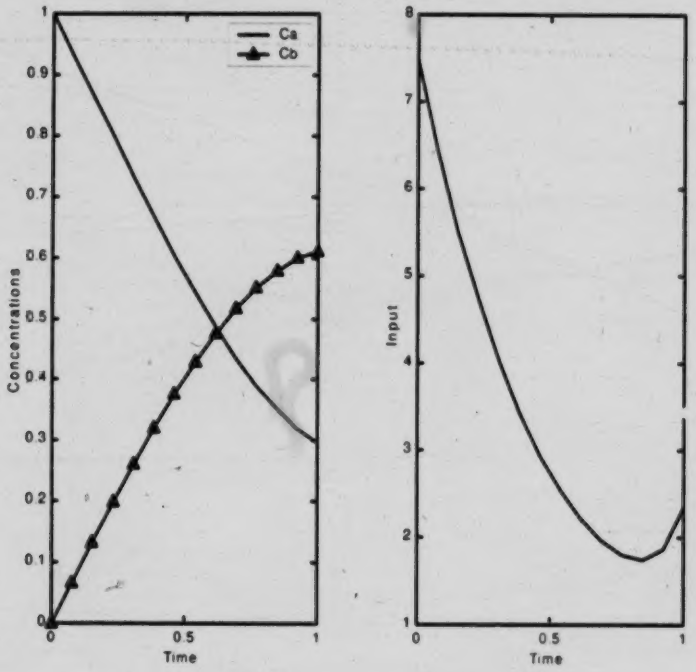


Figure 3.3: Optimal Trajectories for the Consecutive Reaction Problem.

3.4.4 Crane Container Problem

In this complex problem, containers are to be optimally transferred from a ship to a cargo truck using a crane. This problem has posed significant challenges in the optimal control literature and has been considered in Sakawa and Shindo [1982], Goh and Teo [1988], Luus [1991], Teo *et al.* [1991] and Dadebo and McAuley [1995]. The optimization problem for

this system can be represented as:

$$
\begin{aligned}
&\min_{u(t)} \quad \frac{9}{2}\int_0^1 \left[x_3^2(t) + x_6^2(t)\right] dt \\
&s.t.: \\
&\qquad \dot{x}_1(t) = 9x_4(t) \\
&\qquad \dot{x}_2(t) = 9x_5(t) \\
&\qquad \dot{x}_3(t) = 9x_6(t) \\
&\qquad \dot{x}_4(t) = 9[u_1(t) + 17.25656x_3(t)] \\
&\qquad \dot{x}_5(t) = 9u_2(t) \\
&\qquad \dot{x}_6(t) = \frac{-9[u_1(t) + 27.0756x_3(t)]}{x_2(t)} - \frac{18x_5(t)x_6(t)}{x_2(t)} \\
&\qquad \mathbf{x}(0) = [0 \quad 22 \quad 0 \quad 0 \quad -1 \quad 0]^T \\
&\qquad \mathbf{x}(1) = [10 \quad 14 \quad 0 \quad 2.5 \quad 0 \quad 0]^T \\
&\qquad -2.834 \le u_1(t) \le 2.834 \\
&\qquad -0.809 \le u_2(t) \le 0.713 \\
&\qquad |x_4(t)| \le 2.5 \qquad\qquad |x_5(t)| \le 1.0
\end{aligned}
\tag{3.55}
$$

where $u_1(t)$ and $u_2(t)$ are the torque of the hoist and the trolley drive motors, respectively. The objective function represents the swing of the container during the transfer, and is to be minimized for safety reasons.

This problem is *normalized* with the following choice of *flat outputs*:

$$
\begin{aligned}
y_1(t) &\equiv x_2(t) \\
y_2(t) &\equiv x_3(t)
\end{aligned}
\tag{3.56}
$$

The state and input variables expressed in terms of the *flat outputs* are:

$$
\begin{aligned}
x_5(t) &= \frac{\dot{y}_1(t)}{9} \\
x_6(t) &= \frac{\dot{y}_2(t)}{9} \\
u_1(t) &= -27.0756y_2(t) - \frac{2}{81}\dot{y}_1(t)\dot{y}_2(t) - \frac{1}{81}\dot{y}_1(t)\ddot{y}_2(t) \\
u_2(t) &= \frac{\ddot{y}_1(t)}{81} \\
x_4(t) &= \int_0^t 9\left[u_1(\sigma) + 17.25656x_3(\sigma)\right] d\sigma \\
x_1(t) &= \int_0^t 9x_4(\sigma) d\sigma
\end{aligned}
\tag{3.57}
$$

Note that this system is not *flat*, as defined in Martin and Rouchon [1995]. The system can be considered *partially flat* by concentrating on the subsystem:

$$
\begin{aligned}
&\dot{x}_2(t) = 9x_5(t) \\
&\dot{x}_3(t) = 9x_6(t) \\
&\dot{x}_5(t) = 9u_2(t) \\
&\dot{x}_6(t) = \frac{-9[u_1(t) + 27.0756x_3(t)]}{x_2(t)} \frac{-18x_5(t)x_6(t)}{x_2(t)}
\end{aligned}
$$

which is *flat* and independent of $x_1(t)$ and $x_4(t)$. The state variables $x_1(t)$ and $x_4(t)$ can be recovered by integrating appropriate expressions in the other state and input variables.

In this case, it is still possible to express the optimization problem in an algebraic form by choosing an appropriate parameterization of the output trajectories. For this problem, the two chosen outputs are parameterized as high-order polynomials:

$$\begin{aligned} y_1(t) &\equiv a_0 + a_1 t + a_2 t^2 + \ldots + a_8 t^8 \\ y_2(t) &\equiv b_0 + b_1 t + b_2 t^2 + \ldots + b_8 t^8 \end{aligned} \tag{3.58}$$

Then, the integrals for $x_1(t)$ and $x_4(t)$ can be solved analytically by symbolic computation.

To solve this problem, the coefficients can be collected into the vector $\boldsymbol{\theta} \equiv [a_0, a_1, \ldots, a_8, b_0, b_1, \ldots, b_8]^T$. From the initial and final time constraints, $a_0 = 22$, $a_1 = -9$, $b_0 = 0$, $b_1 = 0$ to ensure that $x_1(0) = 22$, $x_5(0) = -1$, $x_3(0) = 0$, $x_6(0) = 0$. The dynamic optimization problem is transformed to a semi-infinite optimization problem in the general form:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{min} \quad & J(\boldsymbol{\theta}) \\ s.t.: \quad & \\ & w(\boldsymbol{\theta}) = 0 \\ & p(\boldsymbol{\theta}, t) \leq 0 \\ & t \in [0, 1] \end{aligned} \tag{3.59}$$

where the objective function: $J(\boldsymbol{\theta})$ is a quadratic ($\boldsymbol{\theta}^T \mathbf{H} \boldsymbol{\theta}$) that can be simplified to:

$$\begin{aligned} J(\boldsymbol{\theta}) \;=\; & \frac{53}{45}b_5b_4 + \frac{502}{891}b_5^2 + \frac{13}{12}b_5b_6 + \frac{1346}{1287}b_5b_7 + \frac{383}{378}b_5b_8 + \frac{9}{7}b_6b_0 + \frac{89}{72}b_6b_1 + \frac{25}{21}b_6b_2 \\ & + \frac{23}{20}b_6b_3 + \frac{331}{297}b_6b_4 + \frac{151}{286}b_6^2 + \frac{65}{63}b_6b_7 + \frac{197}{195}b_6b_8 + \frac{10}{9}b_7b_1 + \frac{197}{180}b_7b_2 + \frac{320}{297}b_7b_3 \\ & + \frac{298}{585}b_7^2 + \frac{145}{144}b_7b_8 + \frac{905}{891}b_8b_2 + \frac{2303}{4590}b_8^2 + \frac{26}{35}b_3^2 + \frac{61}{60}b_3b_8 + \frac{5}{3}b_2b_3 + \frac{9}{2}b_0^2 + \frac{9}{2}b_0b_1 \\ & + 3b_0b_2 + \frac{9}{4}b_0b_3 + \frac{9}{5}b_0b_4 + \frac{3}{2}b_0b_5 + \frac{9}{8}b_0b_7 + b_0b_8 + \frac{14}{9}{b_1}^2 + \frac{85}{36}b_1b_2 + \frac{86}{45}b_1b_3 \\ & + \frac{29}{18}b_1b_4 + \frac{91}{90}b_1b_8 + \frac{263}{270}b_2^2, \end{aligned}$$

$$w(\boldsymbol{\theta}) = \begin{bmatrix} f_{x_1}(y_1(0), y_2(0), \dot{y}_1(0), \dot{y}_2(0), \ddot{y}_2(0)) \\ y_1(0) - 22 \\ y_2(0) \\ f_{x_4}(y_1(0), y_2(0), \dot{y}_1(0), \dot{y}_2(0), \ddot{y}_2(0)) \\ \frac{\dot{y}_1(0)}{9} + 1 \\ \frac{\dot{y}_2(0)}{9} \\ f_{x_1}(y_1(1), y_2(1), \dot{y}_1(1), \dot{y}_2(1), \ddot{y}_2(1)) - 10 \\ y_1(1) - 14 \\ y_2(1) \\ f_{x_4}(y_1(1), y_2(1), \dot{y}_1(1), \dot{y}_2(1), \ddot{y}_2(1)) - 2.5 \\ \frac{\dot{y}_1(1)}{9} \\ \frac{\dot{y}_2(1)}{9} \end{bmatrix}$$

and:

$$p(\boldsymbol{\theta},t) = \begin{bmatrix} \left(-27.0756y_2(t) - \frac{2}{81}\dot{y}_1(t)\dot{y}_2(t) - \frac{1}{81}\dot{y}_1(t)\ddot{y}_2(t)\right) - 2.834 \\ -\left(-27.0756y_2(t) - \frac{2}{81}\dot{y}_1(t)\dot{y}_2(t) - \frac{1}{81}\dot{y}_1(t)\ddot{y}_2(t)\right) - 2.834 \\ \left(\frac{\dot{y}_1(t)}{81}\right) - 0.713 \\ -\left(\frac{\dot{y}_1(t)}{81}\right) - 0.809 \\ f_{x_4}(y_1(t), y_2(t), \dot{y}_1(t), \dot{y}_2(t), \ddot{y}_2(t)) - 2.5 \\ -f_{x_4}(y_1(t), y_2(t), \dot{y}_1(t), \dot{y}_2(t), \ddot{y}_2(t)) - 2.5 \\ \left(\frac{\dot{y}_1(1)}{9}\right) - 1 \\ -\left(\frac{\dot{y}_1(1)}{9}\right) - 1 \end{bmatrix}$$

Solving this Semi-Infinite Optimization problem yields a value of $J(\boldsymbol{\theta}) = 0.0057644$ for the objective function, which is slightly larger than the one reported in literature (*e.g.*, Goh and Teo [1988]). The optimal coefficient values are:

$$[a_i] = \begin{bmatrix} 22 \\ -9 \\ 0.81203323219291 \\ -8.22718558451132 \\ 28.99533109212470 \\ -30.46132916124124 \\ -18.61265676009112 \\ 47.89170516480613 \\ -19.39789798328008 \end{bmatrix}; \qquad [b_i] = \begin{Bmatrix} 0 \\ 0 \\ -1.48789265701876 \\ 4.52513247820709 \\ -0.65126268499532 \\ -7.62467243615438 \\ -2.15838245360171 \\ 16.09752766200006 \\ -8.70045046140724 \end{Bmatrix} \tag{3.60}$$

These results differ from those reported by Dadebo and McAuley [1995], where relaxations of the final state constraints were used to allow solution of the problem. The lower values of the objective functions reported by other researchers, in general, result from the infeasibility of their solutions. Using the coefficient values obtained given in Equation (3.60), the final states are:

$$\begin{aligned} x_1(1) &= 10.00007224 \\ x_2(1) &= 13.99999999 \\ x_3(1) &= -0.00000055 \\ x_4(1) &= 2.50000312 \\ x_5(1) &= 0 \\ x_6(1) &= -0.00000061 \end{aligned}$$

Further, Figure 3.4 shows that the path constraints were respected. Thus, the proposed method produced a feasible solution to Problem (3.55), which has an objective function value that is very close to the results reported in the literature. Also, the resulting Semi-Infinite Optimization problem was solved using initial values of zero for the coefficients. This result contrasts with other methods where, accurate initial estimates of the entire input trajectories were required to arrive at a satisfactory answer.

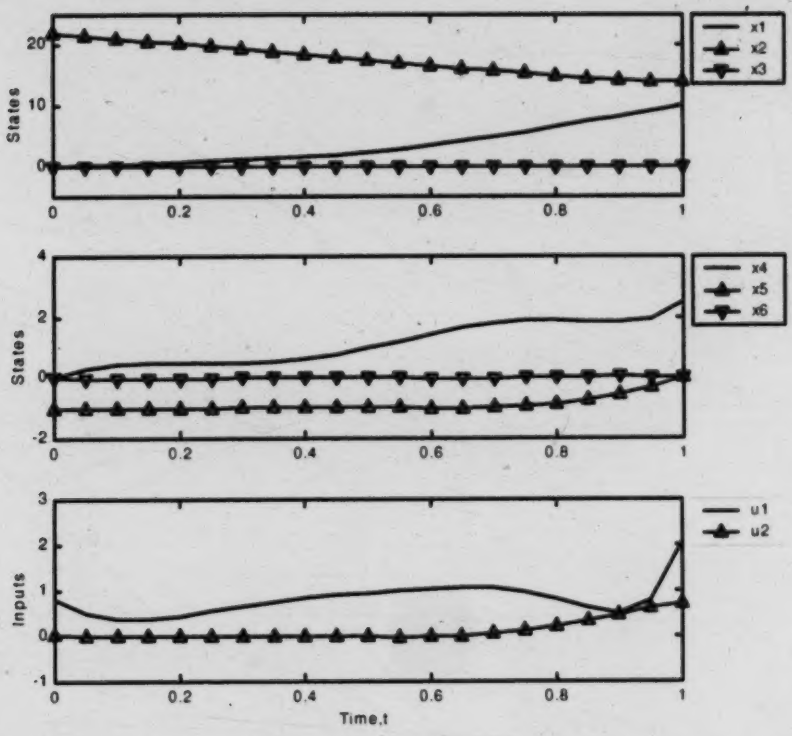


Figure 3.4: Constrained Trajectories for the Crane Container Problem.

3.5 Summary

The results obtained using the proposed NDO approach are compared with those reported in literature are reported in Table 3.1. In general, the proposed method produced similar or better results to those from literature, while being computationally more efficient.

3.6 Conclusions

This chapter presented a new approach for calculating optimal trajectories for *flat* and *partially flat* systems. The proposed approach produced similar or better results than currently available methods based on discretization schemes or dynamic programming for a range of benchmark problems. Although the benchmark problems solved in this chapter were all fixed time problems, it may be possible to extend the method to solve minimum or variable time problems. In the consecutive reaction problem of §3.4.3, a fixed time problem was converted into a variable time problem in the new time scale and solved using the proposed method.

The power of the approach arises from the transformation of a dynamic optimization problem into a simpler and lower-dimensional form. It appears that the method may be

Table 3.1: Comparison of objective function values for illustrative examples.

	Integrator	**Parallel RxN**	**Series RxN**	**Crane**
Proposed Method	0.92423	0.57189	0.610203	0.00576
CVI	—	0.57349[2]	—	—
CVP	0.92518[3]	0.56910[2]	—	0.00540[3]
IDP	0.92428[1] 0.92441[4]	0.57353[1]	0.610775[1]	0.00502[4] 0.00423[1]
Collocation based schemes	—	0.57353[5]	0.610775[2]	—

[1]Dadebo and McAuley[1995]; [2]Biegler [1984]; [3]Goh and Teo [1988]; [4]Luus [1991]; [5]Logsdon and Biegler [1989].

able to produce arbitrary precision for sufficiently smooth problems, with small additional computational cost. Note that the computational complexity for the NDO approach scales with the number of coefficients in the expressions used for representing the *flat outputs* and not the underlying model size or the fineness of the discretization; whereas, in other dynamic optimization methods computational complexity scales with model size, discretization fineness and so forth.

The computational advantages offered by the proposed approach arise because the method exploits the geometry of the system's dynamics to provide co-ordinate transformations, which greatly simplifies the optimization calculation. The determination of these transformations require a substantial analysis stage prior to optimization. Thus, the NDO approach requires more pre- and post-optimization analyses than the conventional approaches, to gain the computational advantages that are promised by the method.

The additional pre-optimization analysis, however, is similar to that required for controller design in a differential geometric framework and the results could be used in formulating the trajectory tracking controller, which would implement the optimization results. Thus, the method seems to be particularly suitable for real-time trajectory generation applications; however, the problem that must be addressed is to ensure that a trajectory tracking controller can be synthesized to enforce the calculated optimal trajectories. This is true, regardless of the method used to determine such trajectories.

Chapter 4

Real-Time Trajectory Optimization

"Never let a computer know you're in a hurry." — Anonymous

4.1 Introduction

Increasing global competition combined with tightening product quality requirements and high operating costs, provides a sufficient economic incentive for the optimal operation of process plants. Among other things such as plant scheduling, operations optimization may require the determination of optimal trajectories for the manipulated variables, such that a desired objective is achieved. Chapter 3 discussed a solution to this problem where a method to determine optimal set-point trajectories was proposed. If the process model and all future disturbances were known perfectly (*i.e.*, if the optimization Problem (3.1) would be applicable, *in toto*, for the duration of the process), then the solution to this problem would lead to an optimal operation of the process. In reality, however, processes are almost always subject to time-varying behaviour or disturbances (*e.g.*, catalyst deactivations, changing stream compositions, *etc.*). In such cases, the originally determined optimal set-point trajectory ceases to be optimal as soon as a disturbance enters the process. In some cases, the trajectory might even become infeasible in the sense that it would violate process constraints (such as valve capacity, coolant availability, *etc.*). Hence for these processes, a scheme is required which is able to update the trajectories on-line so as to remain optimal.

Real-Time Optimization (RTO) is one of the techniques that aims to optimize the steady-state operation of a process [Cutler and Perry, 1983; Darby and White, 1988]. Generally speaking, the RTO system provides the bridge between plant scheduling (which considers long-term inventory, process feeds and product shipments), and the control system (which considers very short-term product quality and process operations safety) [Forbes *et al.*, 1994] (see Figure 4.1).

The purpose of an RTO system is to maintain an economically optimal operations policy for processes with time-varying behaviour[1] [Forbes *et al.*, 1994]. In addition to ensuring the

[1] This chapter provides only a very brief introduction to RTO. For a detailed overview of RTO (the system structure, the major components and applications to process industries), the reader is encouraged to look

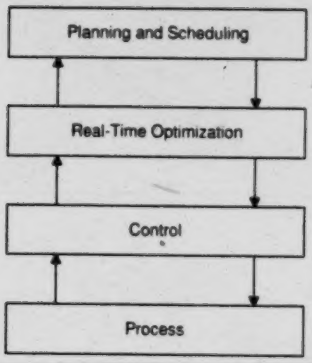


Figure 4.1: Plant Decision-Making Hierarchy.

feasible operation of the process, there are economic benefits to be had when the process is operated so as to compensate for such process changes [Cutler and Perry, 1983]. The degree to which these benefits are realized, however, depends on the design of the RTO system [de Hennin, 1994; Forbes *et al.*, 1994]. Most of the current literature on RTO has focused on what is called model-based steady-state on-line optimization. Such RTO systems are closed-loop, model-predictive control systems that determine the best short-term plant operations policy based on a model and other process information [Forbes, 1994]. The components or subsystems of such an RTO system are: data validation, process model updating, model-based optimization and optimizer command conditioning [Darby and White, 1988]. Figure 4.2 shows this structure.

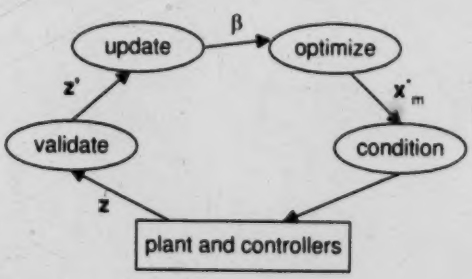


Figure 4.2: Typical model-based RTO system [Forbes, 1994].

The major characteristic of the RTO system depicted in Figure 4.2 is that the plant data ($\mathbf{z}$) is gathered once the plant operation has reached steady-state. This steady-state plant data is validated ($\mathbf{z}'$) to avoid gross errors in the process measurements, and is then used to estimate the model parameters ($\boldsymbol{\beta}$) at the current operating point. Then, the optimum

elsewhere [Cutler and Perry, 1983; Darby and White, 1988; Forbes *et al.*, 1994].

controller setpoints ($\mathbf{x}_m^*$) are calculated using the updated model, and after checking by the command conditioning subsystem, are transferred to the controllers. Such an RTO system has been studied extensively by several researchers [Forbes *et al.*, 1994; Krishnan *et al.*, 1992; Fraleigh *et al.*, 1999; Forbes and Marlin, 1996] and only applies to the currently employed steady-state RTO schemes.

On the other hand, research in the area of Dynamic Real-Time Optimization (DRTO) has been comparatively scarce. DRTO refers to the optimization of processes that are modeled using differential equations and/or algebraic equations with time varying parameters. More specifically, the optimization block in the RTO structure depicted in Figure 4.2 is a dynamic optimization routine, which uses the dynamic process model for optimization (in contrast to a steady-state model used by steady-state optimization routines). The obvious application of this is in the optimization of finite-time processes. Some examples of this type were discussed in Chapter 3.

The major difficulty with the application of DRTO arises because of the very nature of the processes that it can be applied to. Most finite-time processes are modeled by highly nonlinear differential-algebraic equations making trajectory optimization a non-trivial task. Some recent papers have looked at this problem, and provided partial solutions to these problems. Tierno *et al.* [1997] have provided an algorithm that provides a means of evaluating the performance of a nonlinear system in the presence of noise, real parametric uncertainty, and unmodeled dynamics. Nieuwstadt and Murray [1996] have considered the problem of generating a feasible state space and input trajectory in real-time from a reference trajectory. Agrawal *et al.* [1999] proposed a method to determine trajectories of dynamic systems that steer between two end points while satisfying linear inequality constraints arising from limits on states and inputs. Rhee and Speyer [1991;1992] have considered the disturbance attenuation problem over a finite-time interval in the game theoretic approach. In this approach, the control, restricted to a function of the measurement history, plays against adversaries composed of process disturbances and the initial state. One of the few contributions to the dynamic real-time optimization of process control systems is that of Loeblein *et al.* [1999], where operating policies for batch processes subject to parameter uncertainty are determined on-line.

This chapter is concerned with dynamic real-time optimization of finite-time processes, and focuses only on the parameter estimation, model updating and the optimization problems in this framework. The main idea of the chapter is to investigate RTO for dynamic systems using developments of Chapter 3. It must be noted, however, that the discussion to follow is a preliminary attempt of applying NDO algorithm to DRTO, and is not a comprehensive study. The chapter is built up on the scheme proposed by Loeblein *et al.* [1999] for dynamic on-line optimization of batch processes. In the framework proposed by them, the parameter estimation and optimization problem formulations are written as Estimation-Optimization-Tasks (EOTs). The chapter shows the transformation of these formulations

to a *normalized form*, such that the NDO algorithm can be subsequently used to solve the estimation and optimization problems in real-time. This formulation and the solution in *normalized space* is illustrated on three examples of differing complexity.

4.2 Real-Time Optimization of Batch Processes

The most frequently used approach to solve batch process optimization problems is to compute the solution using the nominal process model without taking into account any uncertainties. However, as discussed earlier, when deviations from the nominal model occur (because of disturbances or time-varying model parameters), nominal optimization may no longer be satisfactory. In addition to the nominal model, the objective function formulation then needs to take into account the time-varying characteristics that disturbances introduce to the system. To account for these uncertainties, researchers have primarily looked at the batch optimization problems from two different points of view.

The first approach to the economically optimal and feasible operation of batch processes is based on robust optimization strategies where an uncertain model is assumed for the nominal optimization. In this approach, an optimal operating policy is determined off-line, by optimization of a general objective function of the form: $\Phi(\mathbf{x}, \mathbf{u}, E(\boldsymbol{\eta}))$ that depends on the expected value of the uncertain parameters $\boldsymbol{\eta}$. Related to this approach are: 1) *expected value optimization*, where the expected value of the objective function under uncertainty is optimized, *i.e.*, $E(\Phi(\mathbf{x}, \mathbf{u}, \boldsymbol{\eta}))$ [Ruppen *et al.*, 1995; Terwiesch *et al.*, 1994]; 2) *risk-conscious optimization* [Terwiesch *et al.*, 1994], where the probability of making off-spec product is minimized, or equivalently the probability of satisfying the product constraints is maximized; and 3) *risk threshold optimization* [Terwiesch *et al.*, 1994], where an economic objective function is optimized with a threshold on the risk of violating a constraint. However, all these strategies do not make use of information from the process during its operation, and hence can lead to suboptimal performance when the input profile determined off-line is applied to the time-varying process.

The second approach, dynamic real-time optimization or dynamic on-line optimization, provides an interesting alternative for the optimal operation of these processes. With the assumption that disturbances entering a process are captured by the time-varying parameters of the process model, the off-line optimization method can be applied repeatedly to determine the remaining control trajectory during a batch run. In this approach, the model parameters are updated using the information gathered on-line by measuring one or more process variables. The major difficulty, however, is that the dynamic optimization algorithm used to determine trajectories at every optimization interval needs to be robust as well as computationally fast. This chapter proposes the use of the NDO approach of Chapter 3 for the solution of DRTO problems. This method facilitates the application of one of the many readily available optimization algorithms (*e.g.*, QP, LP, NLP, Semi Infinite Programming, *etc.*) to determine an improved operating policy for the rest of the batch in a receding

horizon manner. This approach may provide improvements over the robust optimization strategies, which in most cases, are suboptimal [Eaton and Rawlings, 1990] . The basic idea of this approach and its similarities to model predictive control (MPC) [Garcia *et al.*, 1989] is summarized in the following section.

4.2.1 MPC and Dynamic Real-Time Optimization

Model predictive control, more commonly referred to as MPC, has been particularly successful for process control applications because of its ability to incorporate constraints directly into the solution of the control law [Garcia *et al.*, 1989]. The basic idea behind MPC [Garcia *et al.*, 1989; Ricker, 1991; Rawlings *et al.*, 1994; and Morari and Lee, 1997] is depicted in Figure 4.3. Using a model of the process, the behaviour of the system with respect to

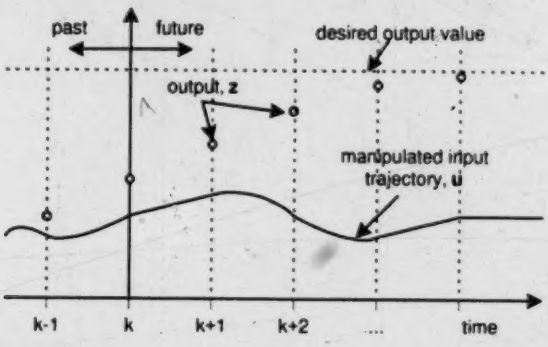


Figure 4.3: Depiction of the General Approach to MPC.

changes in the manipulated variables is predicted at the present time k over some time horizon into the future. Based on this prediction, the manipulated variable moves are determined such that some desired response of the system is achieved as closely as possible. Since MPC is based on the moving time horizon approach, the first part of the calculated input trajectory is applied to the process and the resulting output of the plant is fed back to the controller in order to repeat the procedure at time $k+1$ [Garcia *et al.*, 1989]. The desired response of the system is commonly formulated in terms of a quadratic objective function which penalizes the deviation of the controlled output variables from their set point. Additionally, excessive control actions are prevented by weighting the rate of change of the manipulated variables in the objective function.

The dynamic real-time optimization problem (DRTO) adapts itself particularly well to the above discussed approach to MPC. The DRTO problem can be stated as a form of MPC with the objective function formulation of the desired response and the constraint handling capacity of MPC being directly analogous to the objective function and the differential-algebraic constraints of the DAOP (3.1) respectively. The times k and $k+1$ in MPC are

analogous to the times when the optimization of DAOP is performed in DRTO. The slight difference between MPC and DRTO of batch processes is in the prediction horizon. Batch processes are finite time processes (*i.e.*, their behaviour is only important up to a certain specified final time), and so the time horizon over which the predictions have to be made is shrinking after every optimization interval.

In this framework, Thomas *et al.* [1997] proposed a batch variant of the shrinking-horizon model predictive control (SHMPC) [Joseph and Hanratty, 1993] featuring a control horizon that shrinks with the well-defined batch run end. The SHMPC strategy accepts nonlinear process models, that are corrected on-line using available on-line secondary measurements, to correct for model errors and effects of unmeasured disturbances. The SHMPC strategy involves the use of a NLP based algorithm to solve the updated problem posed at every optimization interval. The paper illustrated the application of the strategy to predict and control the end product quality of composite laminates produced by batch autoclave curing. A similar approach is that of Loeblein [1997], where the concept of Estimation-Optimization-Tasks (EOTs) for formulating the dynamic real-time optimization problem under parametric uncertainty was proposed. This approach, similar to above, involves solving the optimization problem using NLP based techniques. The following section summarizes the approach, and gives the formulations for the estimation and optimization problems proposed in Loeblein [1997].

4.2.2 DRTO - Problem Formulation as EOTs

Loeblein *et al.* [1999] referred to dynamic real-time optimization as the two-step procedure depicted in Figure 4.4. The process is subject to parameter uncertainty and disturbances, and hence an operating policy is determined on-line. In the first step, the process model

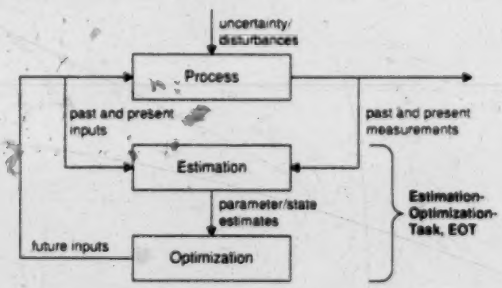


Figure 4.4: Structure of a Dynamic Real-Time Optimizer [Loeblein, 1997].

is identified or updated by estimating the state variables and/or a set of parameters using past and present process measurements. The updated model is then optimized with respect to the manipulated variables and a new optimal input trajectory over the remaining time horizon is determined. The first part of the calculated input trajectory is applied to the

process until the procedure is repeated at the next optimization interval. This sequence of an estimation and optimization step at every optimization interval was referred to as an Estimation-Optimization-Task (EOT)[2] by Loeblein [1997].

In the DRTO framework, the process model used to predict the future behaviour of the plant is assumed to be given by the following modified version of the DAOP (3.1):

$$\begin{aligned} \underset{\mathbf{x}(t),\mathbf{u}(t)}{min} \quad & \Phi(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) && t \in [t_0, t_f] \\ s.t.: \quad & \\ & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) && \mathbf{x}(t_0) = \mathbf{x}_0 \\ & \mathbf{g}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) \leq 0 \\ & \mathbf{c}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) = 0 \\ & \mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U \\ & \mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U \end{aligned} \tag{4.1}$$

where, as in previous chapters, the state and manipulated input trajectories are denoted by $\mathbf{x}$ and $\mathbf{u}$, and $\boldsymbol{\eta}$ is the vector of the time-varying or uncertain parameters in the model[3]. In DRTO, the dynamics of the system $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta})$ are taken into account for the parameter estimation and optimization steps. The optimization results (optimal manipulated variable trajectories) are applied following the concept of moving time horizon shown in Figure 4.3.

In this framework, Loeblein [1997] identified the parameter estimation problem as calculating a set of parameters ($\boldsymbol{\eta}$) at the present time ($\bar{t}$), using the process measurements collected at discrete times over the past time horizon: $\mathbf{z}(\bar{t} - j\Delta T_i),\ j = 0, \ldots, N_i$. Here, $\mathbf{z}$ represents the states that can be measured ($\mathbf{z} \subset \mathbf{x}$), ΔT_i refers to the time difference between two measurements in the past time horizon and N_i is the number of intervals at the present time $\bar{t}$, counted from the start. The parameter estimation problem was then stated as:

$$\left.\begin{aligned} \underset{\boldsymbol{\eta}(t)}{min} \quad & \sum_{j=0}^{N_i} \|\mathbf{z}_{meas}(t_j) - \mathbf{z}_{model}(t_j)\|^2 \\ s.t.: \quad & \\ & \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) \\ & \mathbf{x}_0 = \mathbf{x}(\bar{t} - N_i\Delta T_i) \end{aligned}\right\} \text{Estimation} \tag{4.2}$$

where $\mathbf{z}_{meas}$ and $\mathbf{z}_{model}$ denote the measured and the model values of the states $\mathbf{z}$, and $\mathbf{z}_{model}$ is assumed to be described by the expression: $\mathbf{z}_{model}(t_j) = \mathbf{h}(\mathbf{x}(t_j),\mathbf{u}(t_j),\boldsymbol{\eta})$. The past time horizon t_j is given by the expression: $t_j = \bar{t} - j\Delta T_i$. This differs from steady-state optimization in the sense that the process need not be at steady-state when measurements for parameter estimation are taken.

[2] This thesis uses the definitions and terminology (*e.g.*, the term EOT) used in Loeblein [1997] to frame the real-time optimization problem in the *normalized form*. The difference here is that the NDO algorithm is used to solve the estimation and optimization problems, compared to the NLP based technique proposed in the publication.

[3] This chapter looks at only the case of parametric uncertainty in the models. In other words, the plant-model mismatch is assumed to be completely represented by the uncertain parameter values. The case of structural mismatch of models is beyond the scope of this thesis.

The second step is the dynamic optimization of the updated model to find an input trajectory over the future time horizon which maximizes/minimizes an objective function subject to constraints. This trajectory optimization problem for the DAOP (4.1) can be stated as:

$$\left.\begin{array}{lll} \underset{\mathbf{u}(t)}{min} & \Phi(\mathbf{x}(t_f),\mathbf{u}(t_f),\boldsymbol{\eta}) & t\in[\bar{t},t_f] \\ s.t.: & & \\ & \mathbf{g}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta})\leq 0 & \\ & \mathbf{c}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta})=0 & \\ & \dot{\mathbf{x}}(t)=\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}) & \mathbf{x}(0)=\mathbf{x}(\bar{t}) \\ & \mathbf{x}_L\leq\mathbf{x}(t)\leq\mathbf{x}_U & \\ & \mathbf{u}_L\leq\mathbf{u}(t)\leq\mathbf{u}_U & \end{array}\right\} \text{Optimization} \qquad (4.3)$$

It should be noted that the optimization problem in (4.3) differs from the one proposed in the reference publication. The problem here refers to the calculation of time-varying control trajectories, compared to the piecewise approximations that were sought in Loeblein [1997]. The NDO algorithm facilitates the calculation of trajectories, without the need for piecewise approximations, and hence the EOTs are formulated in this manner.

In this framework, only the first part of the calculated input trajectory is applied to the process initially. Hence, the strategy is similar to the moving or receding horizon principle of MPC. The difference is, however, that the size of the prediction horizon during the on-line optimization of the batch process shrinks at each EOT because its operation is only considered until the final batch time. In the next section, the transformation of these problems to the *normalized space* is presented, and the solution method of the *normalized* problem discussed.

4.3 EOTs in the NDO Framework

This section focuses on the transformations of the EOTs to the *normalized form*, so that the NDO algorithm may be used for the DRTO of problems represented by DAOP (4.1). More specifically, this section shows the transformation of the estimation and optimization Problems (4.2) and (4.3) to the *normalized form*. It must be noted that the problem transformations to follow are valid only under the assumption that the differential equation dynamics: $\dot{\mathbf{x}}=\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\eta}(t))$ of the optimization problem DAOP (4.1) are *flat*. In other words, it is possible to represent the system dynamics as functions of *flat outputs* ($\mathbf{y}(t)$) and a finite number of their derivatives ($\{\dot{\mathbf{y}}(t),\ddot{\mathbf{y}}(t),\ldots,\mathbf{y}^{(m)}(t)\}$).

With this underlying assumption, and collecting the parameters used to parameterize the *flat outputs* in the vector $\boldsymbol{\theta}$, the following transformed expressions for system variables can be written:

$$\mathbf{x}(t) = \mathbf{f}_{\mathbf{x}}(\boldsymbol{\eta},\boldsymbol{\theta},t) \qquad (4.4)$$

$$\mathbf{u}(t) = \mathbf{f}_{\mathbf{u}}(\boldsymbol{\eta},\boldsymbol{\theta},t) \qquad (4.5)$$

The measured vector of states: $\mathbf{z}$ ($\mathbf{z} \subset \mathbf{x}$) can then be specifically represented as:

$$\mathbf{z}(t) = \mathbf{f_z}(\boldsymbol{\eta}, \boldsymbol{\theta}, t) \tag{4.6}$$

The estimation Problem (4.2) can then be equivalently represented in the following *normalized form*:

$$\left.\begin{array}{l} \min\limits_{\eta} \quad \sum\limits_{j=0}^{N_t} \|\mathbf{z}_{meas}(t_j) - \mathbf{z}_{model}(t_j)\|^2 \end{array}\right\} \begin{array}{l}\text{Normalized} \\ \text{Estimation}\end{array} \tag{4.7}$$

where, $\mathbf{z}_{model}(t_j) = \mathbf{f_z}(\boldsymbol{\eta}, \boldsymbol{\theta}, t_j)$ are the values of the measured states calculated from the model at time t_j. The definition of t_j is the same as in the previous section. It must be noted that this formulation does not have the differential state equation constraints of the original estimation problem, the same information being obtained from the transformed algebraic constraint of Equation (4.6). The estimation problem formulation in Equation (4.7) uses the current as well as the previously measured values of the states $\mathbf{z}$, to carry out the estimation in a least squares sense. In this thesis, however, it is assumed that only one of the states is measured, *i.e.*, $dim(\mathbf{z}) = 1$, which is used to estimate only one uncertain parameter, *i.e.*, $dim(\boldsymbol{\eta}) = 1$. Further, in the examples that follow, only the currently measured value of the state is used to back-calculate the uncertain parameter value. With these assumptions, the estimation problem in Equation (4.7) can be simplified to the following form:

$$\left.\begin{array}{l} \min\limits_{\eta} \quad z_{meas}(\bar{t}) - z_{model}(\bar{t}) \end{array}\right\} \begin{array}{l}\text{Normalized} \\ \text{Estimation}\end{array} \tag{4.8}$$

where, as defined earlier, $\bar{t}$ is the present time, and $\mathbf{z}_{model}(\bar{t}) = \mathbf{f_z}(\boldsymbol{\eta}, \boldsymbol{\theta}, \bar{t})$.

With the same assumptions, the *normalized* optimization problem can be stated as:

$$\left.\begin{array}{ll} \min\limits_{\theta} & \Phi(\eta, \boldsymbol{\theta}, t_f) \qquad\qquad t \in [\bar{t}, t_f] \\ s.t.: & \\ & \mathbf{f_g}(\eta, \boldsymbol{\theta}, t) \leq 0 \\ & \mathbf{f_c}(\eta, \boldsymbol{\theta}, t) = 0 \\ & \mathbf{x}_L \leq \mathbf{f_x}(\eta, \boldsymbol{\theta}, t) \leq \mathbf{x}_U \\ & \mathbf{u}_U \leq \mathbf{f_u}(\eta, \boldsymbol{\theta}, t) \leq \mathbf{u}_U \end{array}\right\} \begin{array}{l}\text{Normalized} \\ \text{Optimization}\end{array} \tag{4.9}$$

where, the uncertain parameter η is a scalar and the definitions of other variables are as defined earlier.

The real-time trajectory optimization problem consists of solving the Problems (4.8) and (4.9) at every optimization interval in the receding horizon framework of MPC. For example, at time $t = 0$, the first optimization is carried out with the nominal value of the uncertain parameter, to obtain optimal trajectories for the batch (up to t_f). At the next optimization interval (say t_1), *i.e.* when the measurement for z becomes available, an estimation of η is carried out. This value is used to update the optimization problem, which is used for the determination of input trajectory from t_1 till the end of the batch, t_f. This procedure is repeated until the end of the batch is encountered. The concept is illustrated on three examples in the next section.

4.4 Illustrative Examples

This section illustrates the application of the *normalized* EOTs, formulated in the previous section, to the real-time optimization of three transient processes taken from literature. The method is evaluated in terms of the ability of NDO to perform the estimation and optimization for the time-varying system of equations accurately. The issue of time required for these computations is also addressed. The major assumptions made for the solution to these problems are described in the following section.

4.4.1 Problem Scope and Solution Assumptions

1. The plant-model mismatch is assumed to consist only in one uncertain parameter with no structural mismatch in the model.

2. It is assumed that at least one state, in which the uncertain parameter appears, can be measured. The uncertain parameter is then estimated by a direct comparison of the measured and the model state values at the current time.

3. To simplify the discussion of the solution scheme, the estimation and optimization is carried out at equally distributed optimization intervals.[4] At these points in time, measurement is taken to estimate the uncertain parameter. The process is then optimized using the updated model, and a new optimal input trajectory is determined over the remaining time horizon. This scheme is applied at every optimization interval until the end of the batch is encountered.

4.4.2 Single Integrator System

The following problem is the modified version of the first example presented in Chapter 3. The system consists of a single integrator, and the uncertainty in this model is represented by η (the parameter that will be updated on-line). The modified system is represented as:

$$\begin{array}{lll} \underset{u(t)}{min} & x_2(t_f) & t_f = 1.0 \\ s.t.: & & \\ & \dot{x}_1(t) = u(t) & x_1(0) = 1 \\ & \dot{x}_2(t) = x_1^2(t) + \eta u^2(t) & x_2(0) = 0 \\ & x_1(t_f) = 1 & \end{array} \tag{4.10}$$

The nominal value of η is 1.0, on substitution of which the problem structure of § 3.4.1 is obtained. The parameter estimation and the trajectory optimization steps (EOTs) are first presented in the original system space. The parameter estimation step for this problem, inline with the formulation discussed in § 4.3 and with the assumption that the measurement

[4] The algorithm, however, poses no restrictions to this: different optimization intervals can be chosen at different stages of the optimization without affecting the applicability of the algorithm.

for x_2 is available, can be formulated as:

$$\begin{aligned} \min_{\eta} \quad & x_{2,meas}(t_j) - x_{2,model}(t_j) \\ s.t.: \quad & \\ & \dot{x}_1(t) = u(t) \\ & \dot{x}_2(t) = x_1^2(t) + \eta u^2(t) \\ & x_{1,0} = x_1(\bar{t} - N_i \Delta T_i) \\ & x_{2,0} = x_2(\bar{t} - N_i \Delta T_i) \\ & t_j = \bar{t} - j\Delta T_i. \end{aligned} \tag{4.11}$$

where η is the uncertain parameter, $x_{1,0}$ and $x_{2,0}$ are the states at the present time $\bar{t}$, N_i is the number of discrete time intervals from the start, at which the measurements are being made and ΔT_i is the width of the time interval. For example, in this problem the RTO interval is chosen as 0.1 time units and hence $\Delta T_i = 0.1$, and at the first RTO interval, $N_i = 1$, and so on. Also, $x_{2,meas}(t_j)$ and $x_{2,model}(t_j)$ represent the measured (actual) and predicted (model) value of the state x_2 at time t_j. The trajectory optimization problem in this framework can then be stated as:

$$\begin{aligned} \min_{u(t)} \quad & x_2(1.0) && t \in [\bar{t}, 1] \\ s.t.: \quad & \\ & \dot{x}_1(t) = u(t) && x_1(0) = x_1(\bar{t}) \\ & \dot{x}_2(t) = x_1^2(t) + \eta u^2(t) && x_2(0) = x_2(\bar{t}) \\ & x_1(1.0) = 1 \end{aligned} \tag{4.12}$$

For DRTO, the estimation and optimization problems represented by Problems (4.11) and (4.12) respectively are to be solved at every time interval ($\Delta T_i = 0.1$). It is evident that the solution to both these problems would require a forward integration of the differential equations, unless collocation based schemes are employed to discretize the whole system.

As in § 3.4.1, the *flat output* for the system is chosen to be $y(t) \equiv x_1(t)$ and is parameterized as: $y(t) \equiv a + bt + ct^2 + dt^3 + et^4$. The parameters are collected in a vector $\boldsymbol{\theta} \equiv [a, b, c, d, e]^T$, and with this definition, the system variables can be represented as:

$$\begin{aligned} x_1(t) &\equiv \begin{bmatrix} 1 & t & t^2 & t^3 & t^4 \end{bmatrix} \boldsymbol{\theta} \\ x_2(t) &= \boldsymbol{\theta}^T \mathbf{H}(\eta, t) \boldsymbol{\theta} \\ u(t) &= \begin{bmatrix} 0 & 1 & 2t & 3t^2 & 4t^3 \end{bmatrix} \boldsymbol{\theta} \end{aligned} \tag{4.13}$$

where:

$$\mathbf{H}(\eta, t) = \begin{bmatrix} t & t^2 & \frac{2}{3}t^3 & \frac{1}{2}t^4 & \frac{2}{5}t^5 \\ 0 & \eta t + \frac{1}{3}t^3 & \frac{1}{2}t^4 & 2\eta t^3 & 2\eta t^4 \\ 0 & 2\eta t^2 & \frac{4}{3}\eta t^3 + \frac{1}{5}t^5 & \frac{1}{3}t^6 & \frac{2}{7}t^7 \\ 0 & \frac{2}{5}t^5 & 3\eta t^4 & \frac{9}{5}\eta t^5 + \frac{1}{7}t^7 & \frac{1}{4}t^8 \\ 0 & \frac{1}{3}t^6 & \frac{16}{5}\eta t^5 & 4\eta t^6 & \frac{16}{7}\eta t^7 + \frac{1}{9}t^9 \end{bmatrix}$$

Making use of the relationships in Equation (4.13), the parameter estimation Problem (4.11) is equivalently represented as the following unconstrained optimization problem:

$$\min_{\eta} \quad x_{2,meas}(\bar{t}) - \boldsymbol{\theta}^T \mathbf{H}(\eta, \bar{t}) \boldsymbol{\theta} \quad \Big\} \text{ Estimation Problem} \tag{4.14}$$

The equivalent trajectory optimization formulation in Problem (4.12) can be represented as:

$$\left.\begin{array}{ll} \min\limits_{\theta} & \theta^T \mathbf{H}(\eta = \eta(t_f - N_i \Delta T_i), t = t_f)\theta \\ s.t.: & \\ & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \theta = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ & \begin{bmatrix} 1 & \bar{t} & \bar{t}^2 & \bar{t}^3 & \bar{t}^4 \end{bmatrix} \theta = \tilde{x}_1 \\ & \theta^T \mathbf{H}(\eta = \eta(\bar{t} - N_i \Delta T_i), t = \bar{t})\theta = \tilde{x}_2 \end{array}\right\} \begin{array}{c} \text{Optimization} \\ \text{Problem} \end{array} \quad (4.15)$$

where, $t_f = 1$ and $\tilde{x}_1$,$\tilde{x}_2$ represent the values of the states at time $\bar{t}$ that were calculated (or measured) at the previous optimization interval. These added constraints are required in order to obtain continuous trajectories for the states at the optimization intervals.

The dynamic real-time optimization problem stated above, in the form of estimation and optimization Problems (4.14) and (4.15), is solved using a readily available NLP based optimization code. The optimization interval chosen is 0.1 time units, implying that a total of 9 estimation and optimization solutions are evaluated (at $0, 0.1, 0.2, \ldots, 0.8, 0.9$ time steps). The random disturbance is assumed to be in the form of a normally distributed measurement noise[5] in x_2 with a standard deviation of 0.01.

The results of real-time optimization on this example, *viz.*, the optimization parameters, estimated parameter values for the on-line optimization case, and the nominal and on-line optimized values for the state x_2 are presented in Table 4.1.

Table 4.1: Results of Real-Time Optimization on the Single Integrator System.

Time, $\bar{t}$	Estimated η	ΔT_i	N_i	t_j	$x_{2,optimal}$	$x_{2,nominal}$
0	1.00000	0	0	0	0.0000	0.0000
0.1	0.99973	0.1	0	0	0.1049	0.1095
0.2	1.00068	0.1	1	0, 0.1	0.2239	0.2149
0.3	0.99928	0.1	2	0, 0.1, 0.2	0.2879	0.3038
0.4	0.99979	0.1	3	0, 0.1, ... , 0.3	0.3795	0.3752
0.5	0.99979	0.1	4	0, 0.1, ... , 0.4	0.4590	0.4540
0.6	1.00027	0.1	5	0, 0.1, ... , 0.5	0.5457	0.5449
0.7	1.00113	0.1	6	0, 0.1, ... , 0.6	0.6402	0.6284
0.8	0.99998	0.1	7	0, 0.1, ... , 0.7	0.7126	0.7027
0.9	0.99971	0.1	8	0, 0.1, ... , 0.8	0.8086	0.8249
1.0	–	0.1	9	0, 0.1, ... , 0.9	0.9231	0.9394

In Chapter 3, the true process optimum for this system was found to be 0.9242. Thus, it is clear that the on-line optimization scheme performs better than the nominal in the

[5] The choice of standard deviation of 0.01 for the random measurement error is based on the assumption that it is a standard choice for assuming 1% noise in the measured variable for the evaluation of RTO schemes (see, Loeblein *et al.* [1999]).

presence of disturbances, the deviation from optimum being -0.11 compared to 0.52 for the nominal case.

Figure 4.5 shows the nominal and the on-line optimized (for the disturbance case) input trajectories along with the grid points at which optimization is performed. Using the NDO formulations, the estimation and optimization tasks at the 10 grid points for the above example took approximately 20 seconds, which is an insignificant amount of time even for a reaction scheme of 1 hour (although batch reaction problems may generally be 4-5 hour or even up to 2-3 day schemes).

In the next example, the NDO method is illustrated on a larger scale problem, and is shown to provide a very efficient algorithm to implement RTO on more complex nonlinear systems.

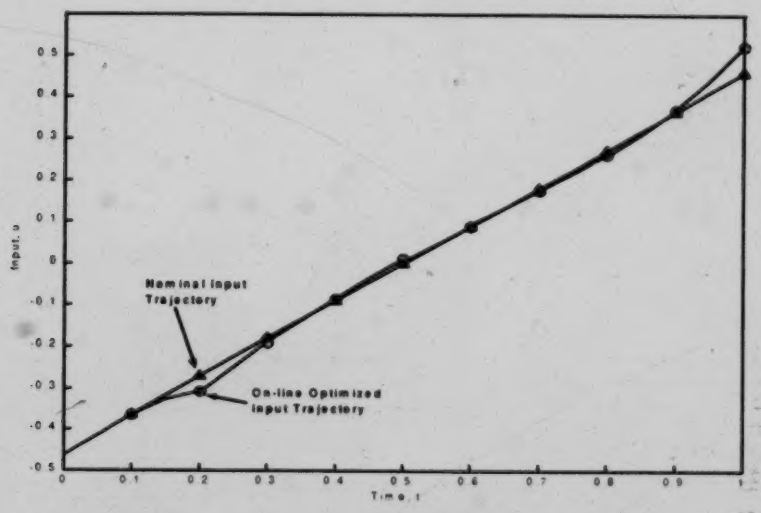


Figure 4.5: Comparison of Input Trajectories for the Single Integrator System.

4.4.3 Crane Container Problem

This example shows the application of dynamic real-time optimization on a larger scale problem. The problem discussed here is the nonlinear crane container system presented in § 3.4.4, the only change being the incorporation of an uncertain parameter η in the state equation for x_4 (the nominal value of η in this case is 17.25656). The optimization problem

for this system can then be represented as:

$$
\begin{aligned}
\min_{u(t)} \quad & \frac{9}{2}\int_0^1 \left[x_3^2(t)+x_6^2(t)\right]dt \\
s.t.: \quad & \\
& \dot{x}_1(t) = 9x_4(t); \qquad \dot{x}_2(t) = 9x_5(t) \\
& \dot{x}_3(t) = 9x_6(t); \qquad \dot{x}_4(t) = 9[u_1(t)+\eta x_3(t)] \\
& \dot{x}_5(t) = 9u_2(t) \\
& \dot{x}_6(t) = \frac{-9[u_1(t)+27.0756x_3(t)]}{x_2(t)} - \frac{18x_5(t)x_6(t)}{x_2(t)} \\
& \mathbf{x}(0) = [0 \quad 22 \quad 0 \quad 0 \quad -1 \quad 0]^T \\
& \mathbf{x}(1) = [10 \quad 14 \quad 0 \quad 2.5 \quad 0 \quad 0]^T \\
& |x_4(t)| \le 2.5 \qquad |x_5(t)| \le 1.0 \\
& -2.834 \le u_1(t) \le 2.834 \\
& -0.809 \le u_2(t) \le 0.713
\end{aligned}
\tag{4.16}
$$

where $u_1(t)$ and $u_2(t)$ are the torque of the hoist motor and the trolley drive motor respectively. The objective function represents the swing of the container during and at the end of the transfer, which has to be minimized for safety reasons. With the same choice of the *flat outputs* ($y_1(t) \equiv x_2(t)$; $y_2(t) \equiv x_3(t)$), the subsequent parameterization:

$$
\begin{aligned}
y_1(t) &\equiv a_0 + a_1 t + a_2 t^2 + \ldots + a_7 t^7 + a_8 t^8 \\
y_2(t) &\equiv b_0 + b_1 t + b_2 t^2 + \ldots + b_7 t^7 + b_8 t^8
\end{aligned}
$$

and the definition: $\boldsymbol{\theta} \equiv [a_0, \ldots a_8, b_0, \ldots, b_8]^T$, the system variables $\{x_1, \ldots, x_6, u_1, u_2\}$ can be written as functions[6] of $\boldsymbol{\theta}, \eta$ and t. Now, with the assumption that the state x_4 is measured, together with the knowledge of the functions representing the system variables (see Chapter 3 for further details), the following *normalized* parameter estimation problem is obtained:

$$
\left.\begin{array}{ll} \min\limits_{\eta} & x_{4,meas}(t_j) - f_{x_4}(\boldsymbol{\theta}, \eta, t_j) \end{array}\right\} \begin{array}{c}\text{Estimation} \\ \text{Problem}\end{array} \tag{4.17}
$$

In the same framework, the corresponding *normalized* optimization problem for this example can be represented as:

$$
\left.\begin{array}{ll} \min\limits_{\boldsymbol{\theta}} & J(\boldsymbol{\theta}) \\ s.t.: & \\ & \mathbf{w}(\boldsymbol{\theta},\eta) = 0 \\ & \mathbf{p}(\boldsymbol{\theta},\eta,t) \le 0 \qquad t \in [\bar{t}, t_f = 1] \\ & \boldsymbol{\mu}(\boldsymbol{\theta}, \eta = \eta(\bar{t} - N_i \Delta T_i), t = \bar{t}) = 0 \end{array}\right\} \begin{array}{c}\text{Optimization} \\ \text{Problem}\end{array} \tag{4.18}
$$

[6]The functions are not stated explicitly here, but can be calculated from the differential equations as shown in Chapter 3. Here, the functional dependence of the variables on $\boldsymbol{\theta}, \eta$ and t will be shown with the relevant subscript to f. For example, the relation for x_1 will be shown as $f_{x_1}(\boldsymbol{\theta}, \eta, t)$ and similarly for others.

where the objective function $J(\boldsymbol{\theta})$ is a quadratic ($\boldsymbol{\theta}^T \mathbf{H} \boldsymbol{\theta}$) that can be simplified to:

$$
\begin{aligned}
J(\boldsymbol{\theta}) \;=\; & \\
& \frac{53}{45}b_5b_4 + \frac{502}{891}b_5^2 + \frac{13}{12}b_5b_6 + \frac{1346}{1287}b_5b_7 + \frac{383}{378}b_5b_8 + \frac{9}{7}b_6b_0 + \frac{89}{72}b_6b_1 + \frac{25}{21}b_6b_2 \\
& + \frac{23}{20}b_6b_3 + \frac{331}{297}b_6b_4 + \frac{151}{286}b_6^2 + \frac{65}{63}b_6b_7 + \frac{197}{195}b_6b_8 + \frac{10}{9}b_7b_1 + \frac{197}{180}b_7b_2 + \frac{320}{297}b_7b_3 \\
& + \frac{298}{585}b_7^2 + \frac{145}{144}b_7b_8 + \frac{905}{891}b_8b_2 + \frac{2303}{4590}b_8^2 + \frac{26}{35}b_3^2 + \frac{61}{60}b_3b_8 + \frac{5}{3}b_2b_3 + \frac{9}{2}b_0^2 + \frac{9}{2}b_0b_1 \\
& + 3b_0b_2 + \frac{9}{4}b_0b_3 + \frac{9}{5}b_0b_4 + \frac{3}{2}b_0b_5 + \frac{9}{8}b_0b_7 + b_0b_8 + \frac{14}{9}{b_1}^2 + \frac{85}{36}b_1b_2 + \frac{86}{45}b_1b_3 \\
& + \frac{29}{18}b_1b_4 + \frac{91}{90}b_1b_8 + \frac{263}{270}b_2^2
\end{aligned}
$$

and constraints have the following form:

$$
w(\boldsymbol{\theta}) = \begin{bmatrix}
f_{x_1}(\boldsymbol{\theta}, \eta, t=0) \\
f_{x_2}(\boldsymbol{\theta}, \eta, t=0) - 22 \\
f_{x_3}(\boldsymbol{\theta}, \eta, t=0) \\
f_{x_4}(\boldsymbol{\theta}, \eta, t=0) \\
f_{x_5}(\boldsymbol{\theta}, \eta, t=0) + 1 \\
f_{x_6}(\boldsymbol{\theta}, \eta, t=0) \\
f_{x_1}(\boldsymbol{\theta}, \eta, t=0) - 10 \\
f_{x_2}(\boldsymbol{\theta}, \eta, t=0) - 14 \\
f_{x_3}(\boldsymbol{\theta}, \eta, t=0) \\
f_{x_4}(\boldsymbol{\theta}, \eta, t=0) - 2.5 \\
f_{x_5}(\boldsymbol{\theta}, \eta, t=0) + 1 \\
f_{x_6}(\boldsymbol{\theta}, \eta, t=0)
\end{bmatrix},
$$

$$
p(\boldsymbol{\theta}, t) = \begin{bmatrix}
f_{u_1}(\boldsymbol{\theta}, \eta, t) - 2.834 \\
-f_{u_1}(\boldsymbol{\theta}, \eta, t) - 2.834 \\
f_{u_2}(\boldsymbol{\theta}, \eta, t) - 0.713 \\
-f_{u_2}(\boldsymbol{\theta}, \eta, t) - 0.809 \\
f_{x_4}(\boldsymbol{\theta}, \eta, t) - 2.5 \\
-f_{x_4}(\boldsymbol{\theta}, \eta, t) - 2.5 \\
f_{x_5}(\boldsymbol{\theta}, \eta, t) - 1 \\
-f_{x_5}(\boldsymbol{\theta}, \eta, t) - 1
\end{bmatrix}
$$

and:

$$
\begin{aligned}
\boldsymbol{\mu}(\boldsymbol{\theta}, \eta \;&=\; \eta(\bar{t} - N_i \Delta T_i), t = \bar{t}) \\
&= \begin{bmatrix}
f_{x_1}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_1 \\
f_{x_2}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_2 \\
f_{x_3}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_3 \\
f_{x_4}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_4 \\
f_{x_5}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_5 \\
f_{x_6}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) - \tilde{x}_6
\end{bmatrix}
\end{aligned}
$$

Here, the notation is the same as in previous example. For this example, the measurement error is assumed to be in x_4, that is normally distributed with a standard deviation of 0.01.

Under the assumptions discussed earlier, the estimation and optimization problems (4.17) and (4.18) are solved at the 10 optimization intervals. The results of real-time optimization on this example, *viz.*, the optimization parameters, estimated parameter values for the on-line optimization case, and the nominal and on-line optimized values for the state x_4 are presented in Table 4.2.

Table 4.2: Results of Real-Time Optimization on the Crane Container Problem.

Time, t	Estimated η	ΔT_i	N_i	t_j	$x_{4,optimal}$	$x_{4,nominal}$
0	17.25656	0	0	0	0.0000	0.0000
0.1	17.25652	0.1	0	0	0.33164	0.32385
0.2	17.25641	0.1	1	0, 0.1	0.41337	0.40865
0.3	17.25570	0.1	2	0, 0.1, 0.2	0.49257	0.48174
0.4	17.25792	0.1	3	0, 0.1, ... , 0.3	0.67042	0.68401
0.5	17.35500	0.1	4	0, 0.1, ... , 0.4	1.01554	1.02523
0.6	17.44096	0.1	5	0, 0.1, ... , 0.5	1.41194	1.41676
0.7	17.40953	0.1	6	0, 0.1, ... , 0.6	1.74297	1.73170
0.8	17.30458	0.1	7	0, 0.1, ... , 0.7	1.89730	1.88591
0.9	17.31318	0.1	8	0, 0.1, ... , 0.8	1.94894	1.95906
1.0	–	0.1	9	0, 0.1, ... , 0.9	2.49436	2.50000

The objective function values obtained for the nominal and on-line optimized cases are 0.005323 and 0.005496 respectively. The optimum for this problem was calculated to be 0.005764. The on-line optimal solution achieves a objective function value that deviates by -0.000268 compared to the -0.000441 deviation achieved by the nominal trajectory. Figure 4.6 shows the nominal and on-line optimized input trajectories obtained by the solution to these problems.

It must be noted in Figure 4.6 that the trajectories for the on-line optimized u_1 and u_2 are not continuous. Physically, the discontinuity in u_2 can be regarded as a sudden change in torque of the trolley drive motor, though undesirable, is physically possible to implement. In practice, care should be taken to ensure smooth transitions between operating regimes.

Using the NDO formulations, the total time for the repeated solution to estimation and optimization problems at the 10 grid points of $t = 0, 0.1, 0.2, \ldots 0.9$ was approximately 100 seconds. This amount of computation time is insignificant for all practical purposes in the operation of the crane. This shows that even for a complex system like above, the computation time may not be an issue with the NDO problem formulation.

4.4.4 Continuous Stirred Tank Reactor

The following continuous stirred tank reactor (CSTR) model comprised of 4 state variables and 2 control inputs, based on realistic physical and chemical assumptions, was proposed by Klatt and Engell [1993]. This reactor model has been further studied by several researchers (*e.g.*, Chen *et al.* [1995]; Klatt *et al.* [1995]; Rothfuss *et al.* [1996] *etc.*). The CSTR is

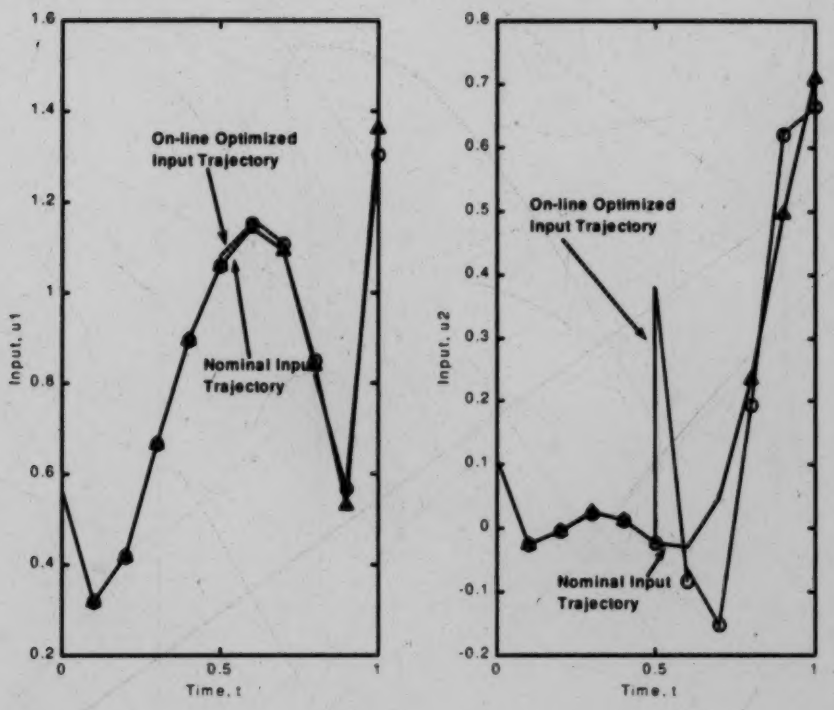


Figure 4.6: Comparison of Input Trajectories for the Crane Container Problem.

comprised of the following material and enthalpy balances:

$$\begin{aligned}
\dot{c}_A &= r_A(c_A,t) + [c_{in} - c_A]u_1 \\
\dot{c}_B &= r_B(c_A,c_B,T) - c_B u_1 \\
\dot{T} &= h(c_A,c_B,T) + \alpha[T_c - T] + [T_{in} - T]u_1 \\
\dot{T}_c &= \beta[T - T_c] + \gamma u_2
\end{aligned} \tag{4.19}$$

where, $c_A(t), c_B(t)$ are the concentration of chemical species A,B and $T(t)$ and $T_c(t)$ are the temperatures in the reactor and cooling jacket respectively. The constraints: $u_1(t) > 0$ and $u_2(t) < 0$ represent the *normalized* flow rate through the reactor and the cooling power applied in the cooling jacket respectively. As given in Klatt and Engell [1993], the inputs are constrained as: $3\ h^{-1} < u_1 < 35\ h^{-1}$ and $-9000\ kJ.h^{-1} < u_2 < 0$. The reaction rates

r_A and r_B and the contribution to enthalpy h due to the reaction are described by:

$$r_A(c_A,t) = -k_1(T)c_A - k_2(T)c_A^2 \quad (4.20)$$

$$r_B(c_A,c_B,t) = k_1(T)\left[c_A - c_B\right] \quad (4.21)$$

$$h(c_A,c_B,T) = -\delta\left[k_1(T)\left[c_A\Delta H_{AB} + c_B\Delta H_{BC}\right] + k_2(T)c_A^2\Delta H_{AD}\right] \quad (4.22)$$

with functions $k_1(T)$ and $k_2(T)$ of the Arrhenius type:

$$k_1(T) = k_{i0}\exp\left(\frac{-E_i}{T/^oC + 273.15}\right), i = 1,2 \quad (4.23)$$

All other symbols denote constant parameters and are tabulated here from the values given in Rothfuss *et al.* [1996].

Table 4.3: Model parameters of the CSTR [Rothfuss et al., 1995]

α	=	$30.828\ h^{-1}$	β	=	$86.688\ h^{-1}$
δ	=	$3.522 \times 10^{-4}\ m^3.K.kJ^{-1}$	γ	=	$0.1\ K.kJ^{-1}$
T_{in}	=	$104.9\ ^0C$	c_{in}	=	$5.1 \times 10^2\ mol.m^{-3}$
k_{10}	=	$1.287 \times 10^{12}\ h^{-1}$	k_{20}	=	$9.043 \times 10^6\ m^3.(mol.h)^{-1}$
E_1	=	9758.3	E_2	=	8560.0
ΔH_{AB}	=	$4.2\ kJ.mol^{-1}$	ΔH_{BC}	=	$-11.0\ kJ.mol^{-1}$
ΔH_{AD}	=	$-41.85\ kJ.mol^{-1}$			

It was shown in the paper by Rothfuss *et al.* [1996] that the above nonlinear model of the CSTR model is ***differentially flat*** with a choice of the following ***flat outputs***:

$$y_1(t) = T \quad (4.24)$$

$$y_2(t) = \frac{c_{in} - c_A}{c_B} \quad (4.25)$$

The system variables were then written as functions of these ***flat outputs*** as[7]:

$$\begin{aligned}
T &= y_1 \\
c_B &= \frac{c_{in}}{y_2} - \frac{k_1(y_1)}{2k_2(y_1)} + \frac{\dot{y}_2 + \sqrt{\left[\dot{y}_2 - k_1(y_1)y_2^2\right]^2 + 4c_{in}y_2k_2(y_1)\left[\dot{y}_2 - k_1(y_1)y_2\right]}}{2y_2^2k_2(y_1)} \\
c_A &= -y_2\frac{k_1(y_1)}{2k_2(y_1)} + \frac{\dot{y}_2 + \sqrt{\left[\dot{y}_2 - k_1(y_1)y_2^2\right]^2 + 4c_{in}y_2k_2(y_1)\left[\dot{y}_2 - k_1(y_1)y_2\right]}}{2y_2k_2(y_1)} \\
T_c &= \psi_1(y_1, y_2, \dot{y}_1, \dot{y}_2, \ddot{y}_2) \\
u_1 &= \psi_2(y_1, y_2, \dot{y}_1, \dot{y}_2, \ddot{y}_2) \\
u_2 &= \psi_3(y_1, y_2, \dot{y}_1, \dot{y}_2, \ddot{y}_1, \ddot{y}_2, \dddot{y}_2)
\end{aligned} \quad (4.26)$$

The control problem for the CSTR studied in the paper by Rothfuss *et al.* [1995] consisted of changing the operation scheme along suitable desired trajectories, such as to obtain a maximal concentration c_B of the intermediate product B. The design of suitable trajectories

[7] The expressions for T_c, u_1 and u_2 are not stated explicity here because of their complexity, but can be calculated using equations (4.19) through (4.25).

was based on the explicit stationary solution of the problem, *i.e.*, the one corresponding to $c_{B,S} : \dot{c}_B = 0$. The stationary concentration $c_{B,S}$ was calculated explicitly as a unique function of the stationary *flat outputs* (*i.e.*, the one corresponding to $\dot{\mathbf{y}} = 0$):

$$c_{B,S} = \frac{c_{in}}{y_2} - \frac{k_1(y_1)}{2k_2(y_1)} + \frac{\sqrt{k_1^2(y_1)y_2^2 - 4c_{in}k_1(y_1)k_2(y_1)}}{2y_2k_2(y_1)}, y_2 \neq 0 \tag{4.27}$$

The operation scheme considered in their paper was: starting from a stationary point $\mathbf{y}_S^a$, while keeping the temperature $T = y_1$ constant, y_2 was increased in order to reach the maximal $c_{B,S}$ corresponding to $\mathbf{y}_S^b$. Then the temperature was increased while staying at the maximal concentration to reach the steady concentration corresponding to $\mathbf{y}_S^c$. They argued that such an operation may be of practical interest, and studied the resulting control problem (see Rothfuss *et al.* [1995] for further details).

In their discussion, c_{in} was identified to be uncertain, but was assumed to be a constant parameter with the assumption that it was varying very slowly. The optimization problem that is considered in this thesis, however, will take into account this uncertainty to analyze the applicability of the proposed real-time optimization scheme. The optimization problem considered is to find the optimal trajectories that take the system under uncertainty from one steady-state to another in a fixed amount of time. More specifically, the optimization problem can be described as: starting from a steady-state concentration of c_B, $c_{B,S} = 1007.12\ mol.m^{-3}$, optimal trajectories are sought that can steer the system to the maximal possible steady-state concentration of c_B in a specified amount of time. Such an optimization problem may be of practical interest, for *e.g.*, in grade change transitions in a multi-product plant. Further, the inflow concentration, c_{in}, is assumed to be uncertain and the random disturbance introduced by this uncertainty is assumed to be in the form of randomly distributed measurement noise in c_A with a standard deviation of 100. In the following, c_{in} is represented by the uncertain parameter, η in the problem formulations. The nominal value for this parameter (from Table 4.3) is $5100\ mol.m^{-3}$.

The above discussed optimization problem can be formulated as:

$$\begin{array}{lll}
\underset{\mathbf{u}(t)}{max} & c_B(t_f = 0.5) & t \in [0, t_f = 0.5] \\
s.t.: & & \\
& \dot{c}_A = r_A(c_A, t) + [\eta - c_A]u_1 & c_A(0) = 2884.33 \\
& \dot{c}_B = r_B(c_A, c_B, T) - c_B u_1 & c_B(0) = 1007.12 \\
& \dot{T} = h(c_A, c_B, T) + \alpha[T_c - T] + [T_{in} - T]u_1 & T(0) = 110 \\
& \dot{T}_c = \beta[T - T_c] + \gamma u_2 & T_c(0) = 105.54 \\
& 3 < u_1 < 35 \qquad -9000 < u_2 < 0 &
\end{array} \tag{4.28}$$

where, the reaction rate functions, r_A, r_B and h are described by the Equations (4.20) through (4.22), and the parameter values are given in Table 4.3. As discussed above, the differential equation model for this system is flat. The *flat outputs* for the system were given in Equations (4.24) and (4.25). Here, they are parameterized using the following

polynomials:

$$y_1 \equiv T \equiv a_0 + a_1 t + a_2 t^2 \tag{4.29}$$

$$y_2 \equiv \frac{c_{in} - c_A}{c_B} \equiv b_0 + b_1 t + b_2 t^2 \tag{4.30}$$

The optimization parameters are collected in a vector $\boldsymbol{\theta} = [a_0, \ldots, a_2, b_0, \ldots, b_2]^T$. Now using Equations (4.29), (4.30) and the expressions in Equation (4.26), together with the knowledge that c_{in} is a uncertain parameter represented by η, the system variables can be written as the following functions:

$$\begin{array}{l} c_A = f_{c_A}(\boldsymbol{\theta}, \eta, t) \\ c_B = f_{c_B}(\boldsymbol{\theta}, \eta, t) \\ T = f_T(\boldsymbol{\theta}, \eta, t) \\ T_c = f_{T_c}(\boldsymbol{\theta}, \eta, t) \\ u_1 = f_{u_1}(\boldsymbol{\theta}, \eta, t) \\ u_2 = f_{u_2}(\boldsymbol{\theta}, \eta, t) \end{array} \tag{4.31}$$

In a real-time framework, the optimization Problem (4.28), using transformations in Equation (4.31), can be formulated as the following *normalized* estimation and optimization problems:

$$\left.\begin{array}{l} \min\limits_{\eta} \quad c_{A,meas}(t_j) - f_{c_A}(\boldsymbol{\theta}, \eta, t_j) \end{array}\right\} \text{ Estimation} \tag{4.32}$$

and:

$$\left.\begin{array}{ll} \max\limits_{\mathbf{u}(t)} & f_{c_B}(\boldsymbol{\theta}, \eta, t_f = 0.5) \\ s.t.: & \\ & f_{c_A}(\boldsymbol{\theta}, \eta, t = 0) = 2884.33 \\ & f_{c_B}(\boldsymbol{\theta}, \eta, t = 0) = 1007.12 \\ & f_T(\boldsymbol{\theta}, \eta, t = 0) = 110 \\ & f_{T_c}(\boldsymbol{\theta}, \eta, t = 0) = 105.54 \\ & f_{c_A}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) = \tilde{c}_A \\ & f_{c_B}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) = \tilde{c}_B \\ & f_T(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) = \tilde{T} \\ & f_{T_c}(\boldsymbol{\theta}, \eta(\bar{t} - N_i \Delta T_i), \bar{t}) = \tilde{T}_c \\ & 3 < f_{u_1}(\boldsymbol{\theta}, \eta, t) < 35 \\ & -9000 < f_{u_2}(\boldsymbol{\theta}, \eta, t) < 0 \\ & t \in [\bar{t}, t_f = 0.5] \end{array}\right\} \text{ Optimization} \tag{4.33}$$

The Optimization Problem (4.33) is solved using a Semi-Infinite Optimization scheme. The optimization of the system at the 5 grid points $(0, 0.1, \ldots, 0.4)$ took approximately 50 seconds. The results are given in Table 4.4.

The solution obtained by the on-line optimization scheme compares favourably with the one obtained from the nominal scheme, the objective function values being $c_{Bf_{optimal}} = 1254.68\ mol.m^{-3}$ and $c_{Bf_{nominal}} = 1236.30\ mol.m^{-3}$, respectively. Figures 4.7 and 4.8 depict the comparison of nominal and on-line optimized input and state trajectories obtained from the solution.

Table 4.4: Results of Real-Time Optimization on the CSTR System.

Optimization Time, $\bar{t}$	Estimated η	ΔT_i	N_i	t_j	$c_{A,nominal}$	$c_{A,optimal}$
0	5100.00	0	0	0	2884.33	2884.33
0.1	5127.78	0.1	0	0	2673.73	2673.73
0.2	5106.90	0.1	1	0, 0.1	2400.02	2444.14
0.3	5453.36	0.1	2	0, 0.1, 0.2	2300.61	2337.43
0.4	5747.16	0.1	3	0, 0.1, ... , 0.3	1743.21	1889.01
0.5	–	0.1	4	0, 0.1, ... , 0.4	985.54	1221.83

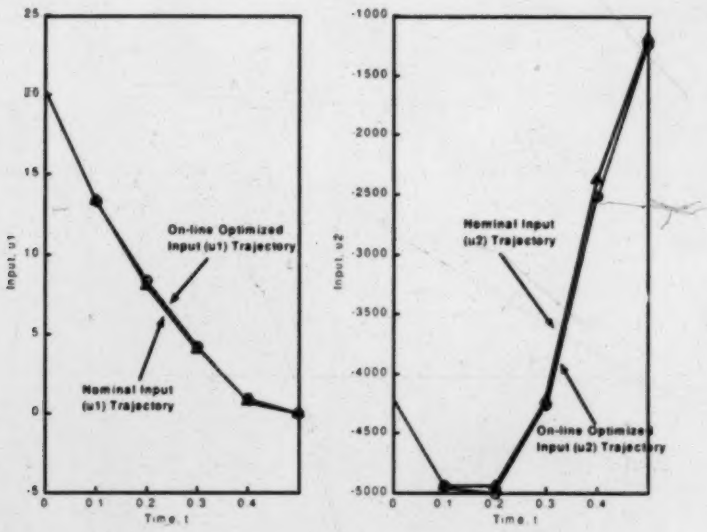


Figure 4.7: Comparison of Input Trajectories for the CSTR System.

4.5 Summary of Results

This chapter presented the preliminary investigations performed for the dynamic real-time optimization using the NDO algorithm presented in Chapter 3. In the framework of EOTs for DRTO proposed by Loeblein [1997], the estimation and optimization problems are formulated in the *normalized form*. These NDO suited problem formulations allow easy and fast computations of the DAOP. Using this technique, the on-line optimal input profiles at every discrete time interval are calculated as continuously varying trajectories. This is a major improvement in terms of accuracy from the approximated piecewise constant control actions chosen in other optimization techniques. The potential of this technique has been illustrated on three examples of widely differing complexity.

The results reported in this chapter show considerable promise in the application of

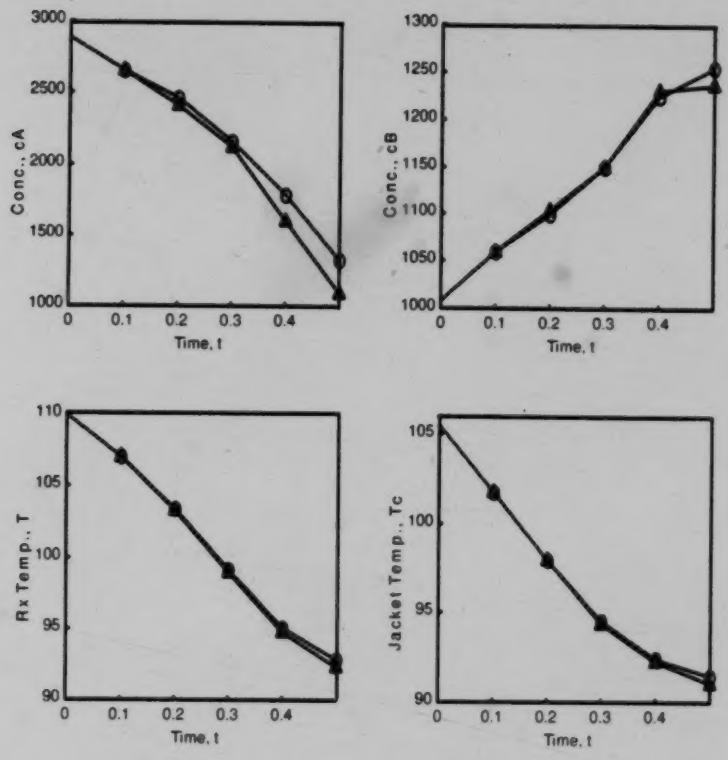


Figure 4.8: Comparison of State Trajectories for the CSTR System.

NDO to complex but *flat* nonlinear systems, where other techniques might fail because of the computational load involved. However, there are several issues which need to be studied in greater detail. First and foremost is the applicability of NDO in a DRTO framework, for which the basic structure of the DRTO system has to be understood properly. This is true, independent of the choice of the algorithm used for estimation and optimization. The design decisions associated with steady-state optimization in addition to the interaction among these components, *viz.*, data validation, parameter estimation, command conditioning, *etc.*, need to be adapted to the dynamic case. One of the major hurdles with DRTO of complex nonlinear systems has always been the large amount of computation time required to solve these problems. In this direction, a new and efficient approach to solve these problems is proposed. It is the hope that with a detailed understanding of the DRTO structure and the interaction among the subsystems, this algorithm will permit the application of DRTO in the batch process industry. Some research directions have been identified in this context, and are discussed in the next chapter.

Chapter 5

Summary and Research Directions

It isn't that they can't see the solution. It is that they can't see the problem. — G. K. Chesterton

5.1 Summary of Results

The optimization of plant operations has been, and continues to be, of considerable interest to academia and industry. The majority of developments in this area have focused on determining the optimal steady-state operation of continuous processes, with little research in the area of dynamic optimization of transient processes. The key characteristic of transient processes is the presence of differential equations in the models. These problems are hence more complex compared to the algebraic optimization problems posed by steady-state processes. The presence of nonlinearities further aggravates the complexity of the problems. The problems rarely possess an analytical solution and computationally expensive numerical solution techniques based on discretization schemes have to be employed to solve them.

This thesis recognized the need for a more efficient and accurate method to solve the dynamic optimization problems posed by these processes. It proposed an alternative method that is applicable to processes whose nonlinear process model is *flat*. Borrowing the concept of *flatness* and *dynamic time scaling* from the nonlinear systems theory, the method transforms the general dynamic optimization Problem (1.9) into an algebraic optimization problem (*i.e.*, LP, QP or NLP) when no path constraints exist, and to a Semi-Infinite Optimization problem in the presence of path and input constraints. In other words, the method permits the transformation of differential equations into algebraic ones, hence reducing or eliminating the number of equations that need to be integrated to solve the optimization problem. This addresses the important issue of computation time for these problems. Moreover, it was shown that the issue of problem complexity (*i.e.*, the number of decision variables) is also addressed properly in such a formulation. The proposed approach produced similar or better results than currently available methods based on discretization schemes or dynamic programming for a range of benchmark problems.

The power of the approach arises from the transformation of a dynamic optimization problem into a simpler and lower-dimensional form. The computational complexity for the NDO approach scales with the number of coefficients in the expressions used for representing the *flat outputs* and not the underlying model size or the fineness of the discretization; whereas, in other dynamic optimization methods computational complexity scales with model size, discretization fineness and so forth. The computational advantages offered by the proposed approach arise because the method exploits the geometry of the system's dynamics to provide co-ordinate transformations, which greatly simplifies the optimization calculation. The determination of these transformations, however, requires a substantial analysis stage prior to optimization. The additional pre-optimization analysis, however, is similar to that required for controller design in a differential geometric framework and the results could be used in formulating the trajectory tracking controller, which would implement the optimization results.

In some cases, the presently employed numerical methods for the solution to such problems guarantee convergence and accuracy. However, they suffer from a drawback that the problem formulations are composed of many decision variables, especially for large nonlinear dynamic systems. This makes the solution computation time a matter of major concern, and hence none of these techniques may be implemented in real-time without some modifications (except maybe for the simplest of problems). In view of the underlying simplicity of the transformed optimization problem, the thesis also investigated the applicability of the proposed method in a real-time framework. In addition to the fast computations that the algorithm allowed, the on-line optimal input profiles at every discrete time interval were calculated as piecewise continuous trajectories. This is a major improvement in terms of accuracy from the approximated piecewise constant control actions chosen by other optimization techniques.

The results reported in Chapters 3 and 4 show considerable promise in the application of NDO to complex but *flat* nonlinear systems, where other techniques might fail because of the computational load involved. However, there are several issues that need to be studied in greater detail. Some of these issues are presented in the next section.

5.2 Future Research Directions

The *normalized dynamic optimization* (NDO) algorithm proposed in this thesis paves way for two major research directions: 1) a more detailed analysis of the algorithm and its applicability; and 2) the issues associated with the real-time formulation of the *normalized* optimization problem. Each of these areas are discussed in the following sections.

5.2.1 NDO Algorithm Issues

The following issues are a direct result of the way the algorithm formulates the optimization problem in the transformed coordinates. The issues need to be analyzed in greater detail

to make the method viable as well as more widely applicable.

Parameterization of Flat Outputs

In this thesis, low order polynomial structures were used to parameterize the *flat outputs*. The results obtained using these simple structures were sufficiently accurate for the illustrative examples. However, both the choice of basis functions as well as the approximation order needs to be carefully analyzed. In this direction, the effect of choosing other kinds of *flat output* parameterizations (*e.g.*, by Legendre polynomials, *etc.*), and their corresponding convergence properties, need to be addressed as well.

Extension to Non-Flat Systems

In §3.4.4., the NDO algorithm was used to solve the optimization problem for the crane container system that was partially *flat*. It was shown, however, that the algorithm was still applicable to the problem and yielded excellent results. The applicability of the algorithm needs to be analyzed for such *non-flat* systems. With a careful analysis, the method may (with or without modifications) be applicable to a larger class of nonlinear dynamic optimization problems.

Free Final-Time Problems

The benchmark problems solved in this thesis were all fixed final-time problems. However, the method (with some minor modifications) may be applied to free final-time optimization problems (*e.g.*, processes that need to minimize the reaction time, *etc.*). The solution scheme of the consecutive reaction problem in §3.4.3 provided an example of this concept. In the example, a fixed time problem was converted into a variable time problem in the new time scale and solved using the proposed method. The method, however, requires a much more careful analysis, before it can be applied to any general case.

5.2.2 Real-Time Implementation Issues

The real-time implementation of the proposed NDO algorithm is dependent on a better understanding of the interaction of the subcomponents of the RTO loop. The research in this area has been relatively scarce, and it is expected that a efficient algorithm such as the proposed one, should provide renewed interest in this area. Some of the issues that are identified in this area are presented in the following sections.

Measurements and Uncertain Parameters

The real-time optimization examples of Chapter 4 assumed that the plant-model mismatch consisted only of the uncertain parameters in the model, with no structural mismatch present. This assumption, however, limits the type of processes that can be handled. Further, it was assumed that only one parameter in the model was uncertain, which was

updated using a single state measurement. Future research should look into the problem formulations for the cases where more than one parameters are updated using the corresponding number (or more) of measurements. Also, in a real process, the measurement of the desired state might not be possible (*e.g.*, if the state represents concentration, and there are no analyzers available). In such a case, a secondary state related to this state may be measured, when the estimation problem would have to be changed. This case was not discussed in the thesis, but should provide an interesting subject for further study.

RTO System Structure

A detailed study needs to be performed on the RTO system for the dynamic case, which should look at the interactions among the subcomponents of the loop. Several results for steady-state optimization exist, and these could be extended to the dynamic case. If the chosen path for dynamic RTO is similar to steady-state RTO, the design decisions associated with steady-state optimization in addition to the interaction among these components, *viz.*, data validation, parameter estimation, command conditioning, *etc.*, need to be adapted to the dynamic case.

Characterization of Disturbances

This thesis investigated the applicability of the algorithm with the simplest possible disturbance structure. The disturbances were assumed to be random measurement errors in one of the states. Moreover, the plant-model mismatch was assumed to be only in the uncertain parameter value with no structural mismatch present. Considerable amount of work needs to be done to analyze the problems with other disturbance characterizations.

Trajectory Tracking Controller

The NDO algorithm calculates the optimal continuous trajectories for the problems analyzed in the real-time framework. The problem, however, is to ensure that a trajectory tracking controller can be synthesized to enforce the calculated optimal trajectories. This is an issue with any optimization technique, and needs a detailed analysis.

Nomenclature

Included in the following list are common symbols employed throughout the thesis. Other, more specific symbols are defined as used.

$a, b, c...$	=	Parameters used to describe the dependence of *flat outputs* on time
A, B, C	=	Chemical species or time invariant matrices in Equation (2.3)
$\mathbf{c}$	=	Vector of equality constraint functions
c_i	=	Concentration of chemical species i
$f(x)$	=	Function of x
$\mathbf{f}(x)$	=	Vector of functions of x
f_x	=	Function defining the variable x
F	=	Time dependent component of the objective function
$\mathbf{g}$	=	Vector of inequality constraint functions
g_i	=	i^{th} inequality constraint function
G	=	Final time dependent component of the objective function
$\mathbf{h}$	=	Vector of measured output functions for a system
$\mathbf{H}$	=	Matrix defining the quadratic objective function
I	=	Pffafian system (see Appendix A)
$I^{(i)}$	=	i^{th} derived system of the Pffafian (see Appendix A)
j	=	Past time horizon in the RTO problem formulation
J	=	Function defining the objective function
k_i	=	Rate constant for the i^{th} reaction
m	=	Number of measured outputs for a system
n	=	Number of states for a system
N_i	=	Number of time intervals from start in the RTO problem formulation
p	=	Number of inputs for a system
s	=	Time scaling function
t	=	Time
$\bar{t}$	=	Present optimization time in the RTO problem formulation
T	=	Temperature
$\mathbf{T}_1$	=	Vector of functions transforming states $\mathbf{x}$ to a set of new states, $\boldsymbol{\xi}$
$\mathbf{T}_2$	=	Vector of functions transforming inputs $\mathbf{u}$ to a set of new inputs, $\mathbf{v}$
ΔT_i	=	Time difference between two measurements in the past time horizon
u_i	=	i^{th} input
$\mathbf{u}$	=	Vector of inputs
$\mathcal{U}$	=	Space where inputs u are defined
$\mathbf{v}$	=	Pseudo input vector for the dynamic state feedback
w_i	=	i^{th} generator for a Pfaffian system (see Appendix A)
$\mathbf{x}$	=	Vector of states

$\overline{\mathbf{x}}$ = Collection of vectors, $\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \ldots$
$\overline{x}$ = Measured or model value for the variable x.
y_i = i^{th} *flat* or measured output
$\mathbf{y}$ = Vector of *flat outputs* or measured outputs for a system
$\mathbf{z}$ = Vector of states that can be measured

Symbols

α = Function relating states to the *flat outputs*
β = Function relating inputs to the *flat outputs*
γ = Static state feedback function
δ = Function describing the coordinate transformation for state feedback
η = Vector of uncertain parameters
θ = Vector of parameters used to characterize the *flat output*
ϑ = Error function for the approximate linearization case
κ = Order of the error function
ξ = States in the new coordinates after state feedback
σ = Variable used to represent the state vector in illustrative examples
τ = New time scale
$\Re$ = Space of real numbers
ϕ = Function of states
ψ = Function of states and inputs
Φ = Objective function for optimization problem formulations

Operators

$\mapsto$ = mapping
$\wedge$ = Wedge product operator (see Appendix A)
$\dot{x}$ = First derivative of x with respect to t
$\ddot{x}$ = Second derivative of x with respect to t
$\dddot{x}$ = Third derivative of x with respect to t
$d(x)$ = Exterior derivative of x (see Appendix A)
dim = Dimension
mod = modulo (see Appendix A)
$E(x)$ = Expected value of x
s = Laplace transform operator

Superscripts

^{n} = Number of states in a system
^{p} = Number of inputs in a system
^{r} = Order of polynomial

Subscripts

0	=	Initial conditions
f	=	Final conditions
k	=	k^{th} iteration
i	=	i^{th} variable
L	=	Lower bound on variables
sp	=	Setpoint
U	=	Upper bound on variables

List of Abbreviations

CSTR	=	Continuous Stirred Tank Reactor
CVI	=	Control Vector Iteration
CVP	=	Control Vector Parameterization
DAE	=	Differential-Algebraic Equation
DAOP	=	Differential-Algebraic Optimization Problem
DRTO	=	Dynamic Real-Time Optimization
EOT	=	Estimation Optimization Task
HJB	=	Hamilton-Jacobi-Bellman
GS	=	Gardner and Shadwick
IDP	=	Iterative Dynamic Programming
LP	=	Linear Programming
max	=	maximize
min	=	minimize
MPC	=	Model Predictive Control
NDO	=	Normalized Dynamic Programming
NLP	=	NonLinear Programming
ODE	=	Ordinary Differential Equation
QP	=	Quadratic Programming
PDE	=	Partial Differential Equation
RTO	=	Real-Time Optimization
s.t.	=	subject to

Bibliography

Agrawal, S.K., N.Faiz and R.M. Murray (1999). Feasible trajectories of linear dynamic systems with inequality constraints using higher-order representations. *1999 IFAC World Congress, Beijing, China.*

Al'Brekht, E.G. (1961). On the optimal stabilization of nonlinear systems. *J. Appl. Math. and Mech.* **25**(5), 836–844.

Beard, R.W. and T.W. McLain (1998). Successive galerkin approximation algorithms for nonlinear optimal and robust control. *Int. J. Cont.* **71**(5), 717–743.

Bellman, R. (1957). *Dynamic Programming.* Princeton University Press. Princeton.

Bequette, B.W. (1991). Nonlinear control of chemical processes: A review. *Ind. Eng. Chem. Res.* **30**, 1391–1413.

Biegler, L.T. (1984). Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers Chem. Engng.* **8**(3/4), 243–248.

Brockett, R.W. (1969). *Finite Dimensional Linear Systems.* Wiley. New York.

Brockett, R.W. (1978). Feedback invariants for non-linear systems. *IFAC Congress* **6**, 1115–1120.

Bryant, R.L., S.S. Chern, R.B. Gardner, H.L. Goldschmidt and P.A. Griffiths (1991). *Exterior Differential Systems.* Springer-Verlag, New York.

Cartan, E. (1953). Sur l'intègration de certains systèmes indéterminés d'équations différentielles. *In Euvres Complètes, volume II, Gauthier-Villars* pp. 1169–1174.

Charlet, B., J. Levine and R. Marino (1991). Sufficient conditions for dynamic state feedback linearization. *SIAM J. Cont. Optim.* **29**, 38–57.

Charlet, R., J. Levine and R. Marino (1989). On dynamic feedback linearization. *Systems and Control Letters* **28**, 517–522.

Chen, H., A. Kremling and F. Allgower (1995). Nonlinear predictive control of a cstr benchmark problem. *Proc. 3rd European Control Conf. ECC'95, Roma, Italy* pp. 3247–3252.

Cuthrell, J.E. and L.T. Biegler (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Computers Chem. Engng.* **13**, 49–62.

Cutler, C.R. and R.T. Perry (1983). Real-time optimization with multivariable control is required to maximize profits. *Computers Chem. Engng.* **7**(5), 663–667.

Dadebo, S.A. and K.B. McAuley (1995). Dynamic optimization of constrained chemical engineering problems using dynamic programming. *Computers Chem. Eng.* **19**, 513–525.

Darby, M.L. and D.C. White (1988). On-line optimization of complex process units. *Chem. Eng. Progress, October* pp. 51–59.

de Hennin, S.R. (1994). Structural Decisions in On-line Process Optimization. PhD thesis. University of London.

Dolcetta, I.C. (1983). On discrete approximate solutions of the hamilton-jacobi-bellman of dynamic programming. *Appl. Math. Optim.* **11**, 367–377.

Eaton, J.W. and J.B. Rawlings (1990). Feedback control of nonlinear processes using on-line optimization techniques. *Computers Chem. Engng.* **14**, 469–479.

Fikar, M., M.A. Latifi, F. Fournier and Y. Creff (1998). Control vector parameterization versus iterative dynamic programming in dynamic optimization of a distillation column. *Computers Chem. Engng.* **22**, S625–S628.

Fliess, M., J. Lévine, P. Martin and P. Rouchon (1993). Linéarisation par bouclage dynamique et transformations de lie-backlund. *Comptes Rendus des Seances de l'Academie des Sciences* **317**(10), 981–986.

Fliess, M., J. Lévine, P. Martin and P. Rouchon (1995). Flatness and defect of nonlinear systems: Introductory theory and examples. *Int. J. Cont.* **61**(6), 1327–1361.

Fliess, M., J. Lévine, P. Martin, Francois Ollivier and P. Rouchon (1997). A remark on nonlinear accessibility conditions and infinite prolongations. *Systems and Control Letters* **31**, 77–83.

Forbes, J.F. (1994). Model Structure and Adjustable Parameter Selection for Operations Optimization. PhD thesis. McMaster University.

Forbes, J.F. and T.E. Marlin (1996). Design cost: A systematic approach to technology selection for model-based real-time optimization systems. *Computers Chem. Engng.* **20**(6/7), 717–734.

Forbes, J.F., T.E. Marlin and J.F. MacGregor (1994). Model adequacy requirements for optimizing plant operations. *Computers Chem. Engng.* **18**(6), 497–510.

Fox, J.M., W.J. Schmidt and J.C. Kantor (1984). Comparison sensitivity and feedback control for optimized chemical reactors. *Proc. 1984 Amer. Cont. Conf., San Deigo, CA* pp. 1621–1627.

Fraleigh, L.M. (1999). Optimal sensor selection and parameter estimation for real-time optimization. Master's thesis. University of Alberta, Edmonton.

Garcia, C.E. and M. Morari (1982). Internal model control 1. A unifying review and some new results. *Ind. Eng. Chem. Proc. Des. Dev.* **21**, 308–323.

Garcia, C.E., D.M. Prett and M. Morari (1989). Model predictive control: Theory and practice - A survey. *Automatica* **25**, 335–348.

Gardner, R.B. and W.F. Shadwick (1992). The gs algorithm for exact linearization to brunovsky normal form. *IEEE Trans. Autom. Cont.* **37**(2), 224–230.

Georgakis, C. (1986). On the use of extensive variables in process dynamics and control. *Chem. Eng. Sci.* **41**, 1471–1484.

Goh, C.J. and L.K. Teo (1988). Control parameterization: A unified approach to optimal control problems with general constraints. *Automatica* **24**, 3–18.

Guay, M. (1996). Measurement of Nonlinearity in Chemical Process Control. PhD thesis. Queen's University, Kingston.

Guay, M. (1999). An algorithm for orbital feedback linearization of single-input control affine systems. *Submitted, Systems and Control Letters.*

Guay, M., P.J. McLellan and D.W. Bacon (1997). A condition for dynamic feedback linearization of control-affine nonlinear systems. *Int. J. Cont.* **68**(1), 87–106.

Guzzella, L. and A. Isidori (1993). On approximate linearization of nonlinear control systems. *Int. J. Rob. Nonlin. Cont.* **3**, 261–276.

Henson, M.A. and D.E. Seborg (1989). A unified differential geometric approach to nonlinear process control. *Presented at the 1989 AIChE Annual Meeting, San Francisco, CA.*

Henson, M.A. and D.E. Seborg (1990). A critique of differential geometric control strategies for nonlinear process control. *Presented at the IFAC World Congress, Tallin, Estonia.*

Henson, M.A. and D.E. Seborg (1997). *Nonlinear Process Control.* Prentice Hall. New Jersey.

Hettich, A. and K. Kortanek (1993). Semi-infinite programming: Theory, method and applications. *SIAM Review* **35**, 380–429.

Hicks, G.A. and W.H. Ray (1971). Approximation methods for optimal control synthesis. *Can. J. Chem. Eng.* **49**, 522–528.

Hollerbach, J.M. (1984). Dynamic scaling of manipulator trajectories. *Trans. ASME* **106**(5), 102–106.

Hunt, L.R., R. Su and G. Meyer (1983). Design for multi-input nonlinear systems. In: *Differential Geometric Control Theory* (R.W. Brockett, R.S. Millman and H.J. Sussman, Eds.). pp. 268–298. Birkhauser. Boston.

Hunt, L.R., R. Su and G. Meyer (1987). An overview of nonlinear geometrical methods for process control. In: *Shell Process Control Workshop* (D.M. Prett and M. Morari, Eds.). pp. 225–250. Butterworths. Boston.

Hunt, R. and J. Turi (1993). A new algorithm for constructing approximate transformations for nonlinear systems. *IEEE Trans. Autom. Cont.* **38**, 1553–1556.

Isidori, A. (1989). *Nonlinear Control Systems: An Introduction.* 2nd ed.. Springer-Verlag. Berlin.

Jakubczyk, B. and W. Respondek (1980). On linearization of control systems. *Bull. Acad. Polonaise Sci. Ser. Sci. Math.* **28**, 517–522.

Jones, C.A., E.F. Johnson, L. Lapidus and R.H. Wilhelm (1963). Design of optimum dynamic control systems for nonlinear processes. *Ind. Eng. Chem. Fundamentals* **2**, 81–89.

Jongen, H.Th. and O. Stein (1997). On generic one-parametric semi-infinite optimization. *SIAM J. Optim.* **7**(4), 1103–1137.

Joseph, B. and F.W. Hanratty (1993). Predictive control of quality in a batch manufacturing process using artificial neural network models. *Ind. Eng. Chem. Res.* **32**(9), 1951.

Klatt, K.-U. and S. Engell (1993). Kontinuierlicher mit neben- and folgereaktion, vdi-berichte, nr. 1026. In: *Nichtlineare Regelung - Methoden, Werkzeuge, Anwendungen.* pp. 101–108. VDI-Verlag.

Klatt, K.-U., S. Engell, A. Kremling and F. Allgower (1995). Testbeispiel:ruhrkesselreaktor mit parallel- und folgereaktion. In: *Entwurf nichtlinearer Regelungen* (S. Engell, Ed.). pp. 425–432. Oldenbourg-Verlag.

Kokotovic, P.V., H.K. Khalil and J.O'Reilly (1986). *Singular Perturbation Methods in Control: Analysis and Design.* Academic Press. New York.

Kravaris, C. and C. Chung (1987). Nonlinear state feedback synthesis by global input/output linearization. *AIChE J.* **33**, 592–603.

Kravaris, C. and J.C. Kantor (1990a). Geometric methods for nonlinear process control I. background. *Ind. Eng. Chem. Res.* **29**, 2295–2310.

Kravaris, C. and J.C. Kantor (1990b). Geometric methods for nonlinear process control II. controller synthesis. *Ind. Eng. Chem. Res.* **29**, 2310–2324.

Krener, A.J. (1984). Approximate linearization by state feedback and cordinate change. *Systems and Control Letters* **5**, 181–185.

Krishnan, S., G.W. Barton and J.D. Perkins (1992). Robust parameter estimation in on-line optimization - part 1. methodology and simulated case study. *Computers Chem. Engng.* **16**, 545–562.

Loeblein, C. (1997). Analysis and Structural Design of On-line Process Optimization Systems. PhD thesis. University of London.

Loeblein, C., J.D. Perkins, B. Srinivasan and D. Bonvin (1999). Economic performance analysis in the design of on-line batch optimization systems. *J. Proc. Cont.* **9**, 61–78.

Logsdon, J.S. and L.T. Biegler (1989). Accurate solution of differential algebraic equations. *Ind. Eng. Chem. Res.* **28**, 1628–1639.

Logsdon, J.S. and L.T. Biegler (1992). Decomposition strategies for large-scale dynamic optimization problems. *Chem. Eng. Sci.* **47**(4), 851–864.

Lu, P. (1993). A new nonlinear optimal feedback control law. *Cont. Theory and Adv. Tech.* **9**(4), 947–954.

Lukes, D.L. (1969). Optimal regulation of nonlinear dynamical systems. *SIAM J. Cont. Optim.* **7**(1), 75–100.

Luus, R. (1991). Application of iterative dynamic programming to state constrained optimal control problems. *Hung. J. Ind. Chem.* **19**, 245–254.

Martin, P. (1992). Contribution à l'étude des systèmes diffèrentiellement plats. PhD thesis. École des Mines de Paris.

Martin, P. and P. Rouchon (1995). Any (controllable) driftless system with m inputs and m+2 states is flat. *Proc. IEEE Cont. Decis. Conf., New Orleans, LA.*

Martin, P., S. Devasia and B. Paden (1994). A different look at output tracking - control of a pivtol aircraft. In: *Proc. 33rd IEEE CDC.* pp. 2376–2381.

McLellan, P.J., T.J. Harris and D.W. Bacon (1990). Error trajectory descriptions of nonlinear control designs. *Chem. Eng. Sci.* **5**, 3017–3034.

Michalska, H. and D.Q. Mayne (1993). Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Autom. Control* **38**, 1623–1633.

Morari, M. (1983). Robust stability of systems with integral control. *Proc. 1983 Conf. Decis. Cont., San Antonio, TX* pp. 865–869.

Morari, M. and J.H. Lee (1997). Model predictive control: Past, present and future. *Presented at the PSE'97-ESCAPE-7 Symposium, Trondheim, Norway.*

Murray, R.M. (1999). Geometric approaches to control in the presence of magnitude and rate saturations. *CDS Technical Report 99-001, Caltech, Pasadena, CA.*

Murray, R.M., M. Rathinam and W.M. Sluis (1995). Differential flatness of mechanical control systems. In: *Proc. ASME Intern. Cong. and Exp.*

Nijmeijer, H. and A. van der Schaft (1990). *Nonlinear Dynamical Control Systems.* Springer-Verlag. Berlin.

Polak, E. (1997). *Optimization Algorithms and Consistent Algorithms*. Springer. New York.

Rathinam, M. and R.M. Murray (1999). Differential flatness of two one-forms in arbitrary number of variables. *Systems and Control Letters* **36**, 317–326.

Rathinam, M. and W.M. Sluis (1995). A test for differential flatness by reduction to single input systems. In: *CDS Technical Report - CDS95-018*. California Institute of Technology. Pasadena, California.

Rawlings, J.B., E.S. Meadows and K.R. Muske (1994). Nonlinear model predictive control: A tutorial and survey. *Proc. IFAC Symp. Adv. Control Chem. Processes - ADCHEM'94* pp. 203–214.

Ray, W.H. (1981). *Advanced Process Control*. McGraw-Hill. New York.

Ray, W.H. (1982). New approach to the dynamics of nonlinear systems with implications for process and control system design. In: *Chemical Process Control 2* (D.E. Seborg and T.F. Edgar, Eds.). pp. 246–267. United Engineering Trustees. New York.

Respondek, W. (1998). Orbital feedback linearization of single-input nonlinear control systems. *Proc. IFAC NOLCOS'98, Enschede, The Netherlands* pp. 499–504.

Rhee, I. and J.L. Speyer (1991). A game theoretic approach to a finite-time disturbance attenuation problem. *IEEE Trans. Autom. Control* **36**(9), 1021–1032.

Rhee, I. and J.L. Speyer (1992). Application of a game theoretic controller to a benchmark problem. *J. Guidance, Control and Dynamics* **15**(5), 1076–1081.

Ricker, N.L. (1991). Model predictive control: State of the art. *Proc. 4th Conf. Chem. Process Control - CPC-IV*.

Rippin, D.W.T. (1983). Simulation of single- and multiproduct batch chemical plants for optimal design and operation. *Computers Chem. Engng.* **7**(3), 137–156.

Rothfuss, R., J. Rudolph and M. Zeitz (1996). Flatness based control of a nonlinear chemical reactor model. *Automatica* **32**(10), 1433–1439.

Rouchon, P., M. Fliess, J. Levine and P. Martin (1993). Flatness, motion planning and trailer systems. In: *Proc. 33rd Conf. Decis. Contr.*. San Antonio, TX.

Ruppen, D., C. Benthack and D. Bonvin (1995). Optimization of batch reactor operation under parametric uncertainty - computational aspects. *J. Process Control* **5**(4), 235–240.

Ruppen, D., D. Bonvin and D.W.T. Rippin (1998). Implementation of adaptive optimal operation for a semi-batch reaction system. *Computers Chem. Engng.* **22**(1-2), 185–199.

Sakawa, Y. and Y. Shindo (1982). Optimal control of container cranes. *Automatica* **18**, 257–266.

Sampei, M. and K. Furuta (1986). On time scaling for nonlinear systems: Application to linearization. *IEEE Trans. Autom. Cont.* **AC-31**(5), 459–462.

Seider, W.D., D.D. Brengel and S. Widagdo (1991). Nonlinear analysis in process design - A review. *AIChE J.* **37**, 1–38.

Shadwick, W.F. (1990). Absolute equivalence and dynamic feedback linearization. *Systems and Control Letters* **15**, 35–39.

Shinskey, F.G. (1962). Controls for nonlinear processes. *Chem. Eng. - March 19* pp. 155–158.

Skogestad, S. and M. Morari (1988). Understanding the dynamic behavior of distillation columns. *Ind. Eng. Chem. Res.* **27**, 1848–1862.

Sluis, W.M. (1992). Absolute Equivalence and its Applications to Control Theory. PhD thesis. University of Waterloo, Waterloo, Ontario.

Soroush, M. and K. Kravaris (1993). Optimal design and operation of batch reactors. *Ind. Eng. Chem. Res.* **32**, 866–881.

Tchon, K. (1994). Towards robustness and genericity of dynamic feedback linearization. *J. Math. Syst. Estim. Cont.* **4**, 165–180.

Teo, K.L., C.J. Goh and K.H. Wong (1991). *A Unified Computational Approach to Optimal Control Problems*. Wiley. New York.

Terwiesch, P., M. Agarwal and D.W.T. Rippin (1994). Batch unit optimization with imperfect modelling: A survey. *J. Proc. Cont.* **4**(4), 238–258.

Thomas, M.M., B. Joseph and J.L. Kardos (1997). Batch chemical process quality control applied to curing of composite materials. *AIChE J.* **43**(10), 2535–2545.

Tierno, J.E., R.M. Murray, J. C. Doyle and I.M. Gregory (1997). Numerically efficient robustness analysis of trajectory tracking for nonlinear systems. *J. Guidance, Control, and Dynamics* **20**(4), 640–647.

Uppal, A., W.H. Ray and A.B. Poore (1974). On the dynamic behavior of continuous stirred tank reactors. *Chem. Eng. Sci.* **29**, 967–985.

van Nieuwstadt, M. and R.M. Murray (1995). Approximate trajectory generation for differentially flat systems with zero dynamics. *Proc. 34th IEEE Conf. on Decision and Control, New Orleans, LA* pp. 4224–4230.

van Nieuwstadt, M., M. Rathinam and R.M. Murray (1994). Differential flatness and absolute equivalence of nonlinear control systems. In: *Proc. IEEE Cont. Decis. Conf., December*. pp. 326–333.

van Nieuwstadt, M.J. and R.M. Murray (1996). Real-time trajectory generation for differentially flat systems. *CIT/CDS 96-017 - To appear, Int. J. Robust and Nonl. Control.*

Villadsen, J. and M.L. Michelson (1978). *Solution of Differential Equation Models by Polynomial Approximation*. Prentice-Hall.

Wright, R.A. and C. Kravaris (1990). Nonminimum phase compensation for nonlinear processes. *Presented at the AIChE Annual Meeting, Chicago, IL.*

Appendix A

Language of Exterior Calculus

Why, a four-year-old child could understand this. Someone get me a four-year-old child. — *Groucho Marx*

This appendix gives a brief introduction to the language of exterior algebra and exterior calculus. Control systems are represented as Pfaffian systems in exterior calculus, and this representation forms the basis for the algorithms used to identify the *flat outputs* for this thesis. This appendix is meant to be a starting point for a reader who is interested in understanding the algorithms presented in Appendix B. The material has been summarized from a number of references [Nieuwstadt *et al.*, 1995; Guay 1997], which should be consulted for specific details. Other research publications [Bryant *et al.*, 1991; Rathinam, 1997; Shadwick, 1990; Sluis, 1992] should be referenced for a broader overview of the inter-relationship of exterior calculus and control systems.

A.1 Exterior Algebra

Let V be an n-dimensional vector space[1] of the field of real numbers $\Re$. The **wedge product** between two vectors $\mathbf{v}, \mathbf{w} \in V$ is a non-commutative product which satisfies the following conditions:

$$\begin{array}{ll} (\alpha v_1 + \beta v_2) \wedge \mathbf{w} = \alpha(v_1 \wedge \mathbf{w}) + \beta(v_2 \wedge \mathbf{w}) & \textit{distributivity} \\ \mathbf{v} \wedge (\alpha w_1 + \beta w_2) = \alpha(\mathbf{v} \wedge w_1) + \beta(\mathbf{v} \wedge w_2) & \textit{bilinearity} \\ \mathbf{v} \wedge \mathbf{w} = -\mathbf{w} \wedge \mathbf{v} \quad \Longrightarrow \mathbf{v} \wedge \mathbf{v} \equiv \mathbf{0} & \textit{skew-commutativity} \end{array}$$

The result of the wedge product of p (one-)vectors is called a p-vector. The space of p-vector on V, $\wedge^p(V)$, is given by the set of all wedge products of the form:

$$\sum_i \alpha_i (v_{1i} \wedge v_{2i} \wedge \ldots \wedge v_{pi}), \quad v_{1i}, \ldots, v_{pi} \in V \quad \textit{and} \quad \alpha_i \in \Re.$$

[1] A vector space is a space made up of elements called vectors along with the operations of addition and multiplication by scalars.

The space $\wedge^p(V)$ has dimension $n!/p!(n-p)!$. Let $\{e_1, \ldots, e_n)$ be a basis for V. The $n!/p!(n-p)!$ elements:

$$\{e_{i_1} \wedge \ldots \wedge e_{i_p}\}, \quad 1 \leq i_1 \leq \ldots \leq i_p \leq n$$

form a basis for $\wedge^p(V)$. Let V^* be a covector space of V, where a covector maps $v \in V$ to $\Re$. By defining an exterior algebra on V^*, the space of p-covectors given by $\Omega^p(V) = \wedge^p(V^*)$ is defined. If $\{e_1^*, \ldots, e_n^*)$ is a basis for V^*, then $\{e_{i_1} \wedge \ldots \wedge e_{i_p}\}$, $1 \leq i_1 \leq \ldots \leq i_p \leq n$, is a basis for $\Omega^p(V)$, the exterior algebra on the covector space of V.

A.2 Differentiable Manifolds

An n-dimensional differentiable manifold M is an abstract set of points which has the following properties: 1) the notion of a continuous function is defined on M; 2) M is the union of a collection of open sets U_i. $M \subset \cup_i, i \in A$ where A is an indexing set; 3) for each $i \in A$, there exists a continuous equivalence between U_i and $\Re^n$ (or, there exists a diffeomorphism $\phi_i : U_i \rightarrow \Re^n$ called a coordinate chart of M); and 4) for any intersecting coordinate charts $U_i \cap U_j$, the change of coordinate map expressed as the composition $\phi_i \circ \phi_j^{-1}$ is a smoothly differentiable map. Conceptually, a differentiable manifold is an abstract space that locally looks like a Euclidean space. Using local coordinate representations, familiar operations from differential calculus can be directly applied to abstract manifolds.

At each point $p \in M$, a tangent space T_pM can be attached to M. The differentiable structure (property 4) allows the concept of a tangent space on a manifold to be related to the familiar concept of a tangent space in $\Re^n$. For a point $x \in \Re^n$, the tangent space $T_x\Re^n$ is the set of tangent vectors to $\Re^n$ at x. The natural basis for $T_x\Re^n$ is denoted by:

$$\left\{ \frac{\partial}{\partial x_1}_x, \ldots, \frac{\partial}{\partial x_n}_x \right\}.$$

The corresponding basis for T_pM is given by the set of vectors:

$$\frac{\partial}{\partial \phi_k}_p = \phi_*(p) \frac{\partial}{\partial x_k}_{\phi(p)}$$

where M is a manifold of dimension n, $p \in M$ and the image of $v \in T_{\phi(p)}\Re^n$ in T_pM is defined as:

$$\phi_*(p)v = \frac{\partial \phi^{-1}}{\partial x} \phi(p) v$$

A vector field v on M is a mapping that assigns a tangent vector $v(p) \in T_pM$ to each $p \in M$. A vector field is smooth if for each $p \in M$ there exists a coordinate chart (U, ϕ) around p and smooth functions $v_1, \ldots, v_n$ on M such that for all $\bar{p} \in U, v(\bar{p}) = \sum_{i=1}^n v_i(\bar{p}) \partial / \partial \phi_i|_{\bar{p}}$, *i.e.*, v can be expressed in terms of the basis vector fields with smooth coefficient functions.

A.3 Differential Forms

The cotangent space, T_p^*M, of M at p is defined as the space of linear mappings of T_pM. Its elements are linear maps $\omega : T_pM \to \Re$. The elements of T_p^*M are called cotangent vectors at p. If $\omega \in T_p^*M$, the value of ω at $v \in T_pM$ is denoted by $\omega(v)$. Letting $\{v_1, \ldots, v_n\}$ be a basis of T_pM at p, the unique basis $\{\omega_1, \ldots, \omega_n\}$ which satisfies $\omega_i(v_j) = \delta_{ij}, 1 \leq i, j \leq n$, is called the dual basis of T_p^*M with respect to $\{v_1, \ldots, v_n\}$.

Given a coordinate chart (U, ϕ) around p, the dual basis to $\left\{\frac{\partial}{\partial \phi_1}_p, \ldots, \frac{\partial}{\partial \phi_n}_p\right\}$ is given by: $\left\{d\phi_{1\,p}, \ldots, d\phi_{n\,p}\right\}$. A differential one-form (or covector field) ω on M is a mapping that assigns to each $p \in M$ a cotangent vector $\omega(p) \in T_p^*M$. A differential one-form is smooth if for each $p \in M$ there exists a coordinate chart (U, ϕ) around p and smooth functions $\omega_1, \ldots, \omega_n$ on M such that for all $\bar{p} \in U$, $\omega(\bar{p}) = \sum_{i=1}^n \omega_i(\bar{p}) d\phi_i|_{\bar{p}}$.

A.4 Exterior Derivative and Differential Systems

Let $\omega = \sum_{i_1 < \ldots < i_p} a_{i_1 \ldots i_p} dx_{i_1} \wedge \ldots \wedge dx_{i_p}$ be a smooth p-form on M where $a_{i_1 \ldots i_p}$ are smooth functions of x_{i_j}. The exterior derivative of ω is the $(p+1)$ form:

$$d(\omega) = \sum_{i_1 < \ldots < i_p} da_{i_1 \ldots i_p} dx_{i_1} \wedge \ldots \wedge dx_{i_p}$$

where, $da_{i_1 \ldots i_p} = \sum_{i=1}^n \frac{\partial a_{i_1 \ldots i_p}}{\partial x_i} dx_i$. The operator $d()$ has the following properties:

$$d(\omega_1 + \omega_2) = d(\omega_1) + d(\omega_2)$$
$$d(\omega \wedge v) = d(\omega) \wedge v + (-1)^p (\omega \wedge d(v))$$
$$d(d(w)) = 0$$

An exterior differential system is given by a homogeneous ideal $I \subset \Omega(M)$ that is closed under exterior differentiation. More specifically, I satisfies:

$$\alpha \in I, \beta \in \Omega(M) \Longrightarrow \alpha \wedge \beta \in I$$
$$\alpha \in I, \alpha = \textstyle\sum_i \alpha_i \omega_i, \omega_i \in \Omega(M) \Longrightarrow \omega_i \in I$$
$$\alpha \in I \Longrightarrow d(\alpha) \in I$$

A.5 Pfaffian and Control Systems

An algebra V is called a **ring** if and only if the following conditions are satisfied:

$$v_1, v_2 \in V, v_1 + v_2 = v_2 + v_1$$
$$v_1, v_2, v_3 \in V, v_1 + (v_2 + v_3) = (v_1 + v_2) + v_3$$
$$0 \in V, v \in V, v + 0 = v$$
$$v \in V, -v \in V, v + (-v) = 0$$
$$v_1, v_2, v_3 \in V, v_1(v_2 v_3) = (v_1 v_2) v_3$$

A **commutative ring** is such that:

$$v_1, v_2 \in V, v_1 v_2 = v_2 v_1.$$

A module for the algebra V is a vector space W over the field K along with a binary product $V \times W$ into W mapping $(v,\omega), v \in V, \omega \in W$ to $v\omega \in W$ such that:

$$\begin{gathered} v(\omega_1 + \omega_2) = v\omega_1 + v\omega_2 \\ (v_1 + v_2)\omega = v_1\omega + v_2\omega \\ \alpha(v\omega) = (\alpha v)\omega = v(\alpha\omega) \\ (v_1 v_2)\omega = v_1(v_2\omega) \end{gathered}$$

for all $v \in V, \omega \in W$ and $\alpha \in K$.

A submodule U of a module W is a subspace closed under the composition by elements of the algebra.

Definition 3 ***(Pfaffian System).*** *A Pfaffian system P on a manifold M is a submodule of the module of differential one-forms $\Omega^1(M)$ over the commutative ring of smooth functions $C^\infty(M)$. A set of one-forms $\omega^1,...,\omega^n$, generates a Pfaffian system $P = \{\omega_1,...,\omega_n\} = \{\sum f_k\omega^k | f_k \in C^\infty(M)\}$. A Pfaffian system defines an exterior differential system:*

$$I = \langle \{P, d(P)\} \rangle = \text{ Ideal generated by } P, d(P).$$

Definition 4 ***(Control System).*** *A control system of the form: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},t)$ where $x \in M \subset \Re^n$ and $u \in \Re^p$ defines a Pfaffian system on the manifold $M^* = M \times \Re^p \times \Re$ with local coordinates $(\mathbf{x},\mathbf{u},t)$ generated by the one-forms:*

$$I = \{\omega_1, \ldots \omega_n\} = \{d(x_1) - f_1(\mathbf{x},\mathbf{u},t)d(t), ..., d(x_n) - f_n(\mathbf{x},\mathbf{u},t)d(t)\}$$

The integral curves $c(s) \in M^*$ of the control system are the solutions of: $\omega(c(s))c'(s) = 0, \omega \in P$, where $c'(s)$ is the velocity vector tangent to $c(s)$.

A.6 Integrability and Congruence of Forms

Theorem 5 ***(Frobenius)*** *A $k-$dimensional codistribution $I = span\{\omega_1, \ldots, \omega_k\}$ is locally integrable if and only if there exist one-forms θ_{ij} such that:*

$$d(\omega_i) = \sum_{j=1}^{k} \theta_{ij} \wedge \omega_j.$$

A closed one-form is such that $d(\omega) = 0$.

Let $I = \{\omega^1,...,\omega^n\}$ be an exterior differential system. Two forms, ω and $\xi \in \Omega(M)$ are congruent **modulo** I, written:

$$\omega \equiv \xi mod\ I$$

if there exists $\mu \in I$ such that $\omega = \xi + \mu$. If I is a Pfaffian system then:

$$\eta \equiv \xi mod\ I \Longleftrightarrow \eta = \xi + \sum_i \theta_i \wedge \omega_i$$

for $\theta_i \in \Omega(M)$. It follows that:

$$\eta \equiv 0 mod\ I \Longleftrightarrow \eta \wedge \omega_1 \wedge \ldots \wedge \omega_k = 0.$$

As a result, from Frobenius theorem it follows that a Pfaffian system I is integrable if and only if

$$d(\omega_i) \equiv 0 mod\ I$$

A.7 Derived Systems and Differential Flatness

Definition 6 ***(Derived System).*** *The derived flag of a Pfaffian system I, is a filtering resulting in a sequence of Pfaffian systems such that $I \supset I^{(1)} \supset \ldots \supset I^{(k)}$. The system $I^{(i)}$ is called the i^{th} derived system of I defined by $\left\{I^{(i)} = \gamma \in I^{(i-1)} | d\gamma \equiv 0 \bmod I^{(i-1)}\right\}$.*

Corollary 7 (Bottom Derived System). *If the system is regular, i.e.,*
1) the system and all its derived systems have constant rank; and
2) for each k, the exterior differential system generated by $I^{(k)}$ has a degree 2 part with constant rank,
the derived flag is decreasing, so there will be an N such that $I^{(N)} = I^{(N+1)}$. This $I^{(N)}$ is called the bottom derived system.

Theorem 8 ***(Differential flatness for Single Input Systems).*** *A Pfaffian system I of constant codimension 2 is flat if and only if*
(1) $\dim I^{(i)} = \dim I^{(i-1)} - 1$, for $i = 0, ..., n = dim I$. This implies $I^{(n)} = \{0\}$.
(2) The system $I^{(i)} + span\{dt\}$ is integrable for each $i = 0, ..., n$.

Appendix B

Algorithms for Characterization of Differential Flatness

This appendix gives the relevant details of the algorithms of Rathinam and Sluis [1995], Gardner and Shadwick [1992] and Guay *et al.* [1997] discussed in Chapter 2. The *flat outputs* for the illustrative examples presented in this thesis have been identified using the algorithm of Guay *et al.* [1997]. The application of the algorithm was illustrated on the 4-state, 2-input example in Chapter 3.

B.1 Reduction of System Codimension

This technique was proposed by Rathinam and Sluis [1995] considering the fact that a complete characterization of *flatness* for single input systems (codimension 2 systems) is known and is given by Theorem 1. For high codimension systems, no complete characterization of *flatness* exists, except for some verifiable necessary conditions. A scheme was proposed that can determine whether a system has *flat outputs* of a particular form. For a system with p inputs, the technique is enumerated in the following 3 steps:

Step 1. Guess for $p-1$ *flat outputs* $y_1, ..., y_{p-1}$. This guess often will involve expressing the *flat outputs* as a parameterized family.

Step 2. Set these guessed *flat outputs* to free functions of time: $y_i = Y_i(t), i = 1, ..., p-1$. Solve for (some of) the variables x in terms of the free functions $Y_i(t)$, and substitute them in the system equations. This leads to a system for which Theorem 1 applies. Note that the resulting system is often time dependent. The resulting system, called the reduced system, is under-determined by 1 equation as opposed to p in the case of the original system.

Step 3. Check whether the conditions of Theorem 8 are satisfied. In the case that they are, a *flat output* z for the reduced system can be calculated. In general, this *flat output* will depend on $\{t, x\}$ and the free functions $Y_i(t)$, but in order that z is the final *flat output* for

the original system, it is necessary that $z = h(t, x)$, which might be true for some values of the parameters used in choosing the *flat outputs*. The test fails if for some parameter value the conditions of Theorem 1 are not satisfied and hence the *flat output* z for the system cannot be identified. In this case, the $p-1$ *flat outputs* in Step 1 are guessed again and the technique is applied again to this new set.

B.2 Gardner and Shadwick (GS) Algorithm

This section presents the necessary and sufficient condition for linearizability of nonlinear systems utilized in the Gardner and Shadwick [1992] Algorithm.

The general form of the systems with n states and p inputs considered here have the form:

$$\dot{x} = f(x, u)$$

Associated with this system is a Pfaffian system given by:

$$I = \{d(x_1) - f_1(x, u)d(t), ..., d(x_n) - f_n(x, u)d(t)\}$$

For a system in *Brunovsky normal form*, there exist Kronecker indexes $k_1 \geq k_2 \geq ... \geq k_p$ and independent functions:

$$t, y_{1,1}, ..., y_{1,k_1}, y_{2,1}, ..., y_{2,k_2}, ..., y_{p,1}, ..., y_{p,k_p}, v_1, v_2, ..., v_p$$

which give the following generators for its associated Pfaffian system:

$$\begin{aligned}
w_{1,1} &= d(y_{1,1}) - y_{1,2}d(t), \ ..., \ w_{1,k_1} = d(y_{1,k_1}) - v_1 d(t) \\
w_{2,1} &= d(y_{2,1}) - y_{2,2}d(t), \ ..., \ w_{2,k_2} = d(y_{2,k_2}) - v_2 d(t) \\
&\vdots \\
w_{p,1} &= d(y_{p,1}) - y_{p,2}d(t), \ ..., \ w_{p,k_p} = d(y_{p,k_p}) - v_p d(t)
\end{aligned}$$

The structure equations for these generators are given by:

$$\begin{aligned}
&d(w_{i,j}) = d(t) \wedge w_{i,j+1} \\
&\qquad\qquad\qquad\qquad i = 1, ..., p, \ j = 1, ..., k_i - 1 \\
&d(w_{i,k_i}) = d(t) \wedge d(v_i)
\end{aligned} \tag{B.1}$$

This yields the following structure for the derived flag of this Pfaffian system:

$$\begin{array}{cccccccccc}
w_{1,1} & w_{1,2} & ... & w_{1,k_1} & ... & ... & w_{p,1} & w_{p,2} & ... & w_{p,k_p} \\
. & . & & & & & . & . & & \\
. & . & & & & & . & . & & \\
. & . & & & & & . & w_{p,2} & & \\
. & . & & & & & w_{p,1} & & & \\
. & . & & & & & & & & \\
w_{1,1} & w_{1,2} & & & & & & & & \\
w_{1,1} & & & & & & & & &
\end{array}$$

where the first row is formed by the generators of the system I and each subsequent row j is formed by the generators of the corresponding j^{th} derived system $I^{(j)}$.

For control systems, the equalities (B.1) are modified to:

$$\begin{aligned} d(w_{1,k_1-j}) &= d(t) \wedge w_{1,k_1-j+1} \\ & \qquad\qquad \text{mod } I^{(j+1)} \\ d(w_{m_j,k_{m_j}-j}) &= d(t) \wedge w_{m_j,k_{m_j}-j+1} \end{aligned}$$

where m_j is the number of towers with at least $j+1$ rows.

Using the above generators as initiators of the towers, the new generators are found which establish the congruences. The linearizing coordinates are then simply obtained as the functions multiplying $d(t)$ in each generator.

B.3 Linearization by Endogenous Dynamic Feedback

A nonlinear control-affine system of the form:

$$\dot{x} = f(x) + g(x)u(t) \tag{B.2}$$

where $x \in M \subset R^n$ and $u \in R^p$, defines a Pfaffian system on the manifold $M^* = M \times R^p \times R$ with local coordinates (x, u, t) generated by the one-forms:

$$\begin{aligned} I &= \{\omega_1, \ldots, \omega_n\} \\ &= \left\{ \begin{array}{c} d(x_1) - (f_1(x) + g_1(x)u)\, d(t), \ldots, \\ d(x_n) - (f_n(x) + g_n(x)u)\, d(t) \end{array} \right\}. \end{aligned}$$

Theorem 9 ***Gardner and Shadwick [1992]*** *A control system I is static state feedback linearizable if and only if*

(1) the k^{th} derived system is trivial

(2) I is generated by $w^i_{v_i} (i = 1, ..., p,\ j = 1, ..., v_i)$ that satisfy congruences:

$$\begin{aligned} dw^i_{v_i} &\equiv dt \wedge du_i \ \ mod\ I \\ dw^i_j &\equiv dt \wedge du_{j+1} \ \ mod\ I^{(j)} \end{aligned}$$

When considering dynamic state-feedback linearization, dynamic precompensators are incorporated that have the form:

$$\begin{aligned} \dot{\xi} &= a(x, \xi) + b(x, \xi)v \\ u &= c(x, \xi) + d(x, \xi)v \end{aligned} \tag{B.3}$$

where the precompensator states $\xi \in R^q$ are smooth functions of the system states, inputs and a finite number of their time derivatives ($= \phi(x, u, \dot{u}, \ldots, u^{(\beta)})$). Dynamic feedback linearizability implies that the combined system (B.2) and (B.3) fulfils the conditions of Theorem 9 and is therefore equivalent to a linear controllable form:

$$\begin{aligned} y^{(\nu_1)} &= v_1 \\ &\vdots \\ y^{(\nu_p)} &= v_p \end{aligned}$$

where $\sum_i \nu_i = n + q$ are the controllability indices of the system (B.2),(B.3).

The dynamic precompensator (B.3) proposed by Guay *et al.* [1997] generally have a prescribed structure. It can be constructed from either differentiation of the original system variables or algebraic functions of them, *i.e.*:

$$\begin{aligned} \dot{v}_i &= v_i^{(1)} \\ \dot{v}_i^{(1)} &= v_i^{(2)} \\ &\vdots \\ \dot{v}_i^{(\alpha_i)} &= v_i^{(\alpha_i+1)} \\ v &= \varphi(x,u) \end{aligned} \tag{B.4}$$

for $1 \leq i \leq p$, or where the indices $\{\alpha_1, \ldots, \alpha_p\}$ give the degree of precompensation of each input channel. In its most general form, a dynamic precompensator can be taken from a combination of differentiation and algebraic functions like:

$$\begin{aligned} \dot{u}_i &= u_i^{(1)} \\ \dot{u}_i^{(1)} &= u_i^{(2)} \\ &\vdots \\ \dot{u}_i^{(\beta_i)} &= u_i^{(\beta_i+1)} \text{ for } 1 \leq i \leq p \\ v &= \varphi(x, u_1, ..., u_1^{(\beta_1+1)}, ..., u_p, ..., u_p^{(\beta_p+1)}) \\ \dot{v}_j &= v_j^{(1)} \\ \dot{v}_j^{(1)} &= v_j^{(2)} \\ &\vdots \\ \dot{v}_j^{(\gamma_j)} &= v_j^{(\gamma_j+1)} \text{ for } 1 \leq j \leq p. \end{aligned}$$

Having defined the structure of the precompensator to be used, the conditions for the dynamic feedback linearizability of control systems can be derived. For simplicity, only the first class of precompensators based on pure differentiation of the system inputs or of a static state-feedback transformation are considered. The structure of these precompensators is completely defined by the degree of precompensation of each input channel. Arranging the indices $\{\alpha_1, \ldots, \alpha_p\}$, $\sum_i \alpha_i = q$ such that $\alpha_1 \geq \ldots \geq \alpha_p$, a precompensator is defined by the indices:

$$\{k_1, \ldots, k_1, k_2, \ldots, k_2, \ldots, k_m, \ldots, k_m\}$$

$(k_1 = \alpha_p, \ldots, k_m = \alpha_p)$ with multiplicities

$$\{s_1, \ldots, s_m\}.$$

Next, a filtering of the original Pfaffian system that takes into account the presence of this precompensator is defined.

Definition 10 *Consider a control-affine system with n states and p inputs, its corresponding Pfaffian system, I, and a precompensator based on the nonlinear feedback $v = \varphi(x, u)$ and indices $\{k_1, ..., k_m\}$ with multiplicities $\{s_1, ..., s_m\}$. The first derived system associated with the precompensator, $I^{(1)}$, is given by the set of forms, $w^{(1)}$, which satisfy*

$$dw^{(1)} \equiv 0 \bmod I, dv_1, ..., dv_{p-s_1}.$$

The second derived system associated with the precompensator is given by the set of forms, $w^{(2)}$, which satisfy

$$dw^{(2)} \equiv 0 \bmod I^{(1)}, dv_1, ..., dv_{p-s_1-s_1}$$

if $k_2 - k_1 > 1$, or

$$dw^{(2)} \equiv 0 \bmod I^{(1)}, dv_1, ..., dv_{p-s_1-s_2}$$

if $k_2 - k_1 = 1$. By induction we get

$$dw^{(k_i-k_1+j)} \equiv 0 \bmod I^{(k_i-k_1+j-1)}, dv_1, ..., dv_{p-s_1-...-s_i}$$

for $1 \leq j \leq k_{i+1} - k_i$, $1 \leq i \leq m-1$. The number Q for which $I^{(Q+1)} = I^{(Q)}$ is called the derived length associated with the precompensator.

As pointed in Guay *et al.* [1997] and Sluis [1995], the class of precompensators admissible for dynamic feedback linearization is bounded. There are restrictions on the structure of the precompensator that are necessary for dynamic feedback linearization. The following lemma defines the class of admissible precompensators of this form for dynamic feedback linearization.

Lemma 11 *Consider a control-affine system with n states and p inputs and define a precompensator by ordering the inputs according to the indices $\{k_1, ..., k_m\}$ with multiplicities $\{s_1, ..., s_m\}$. Dynamic feedback linearization requires that $n - (k_m - k_1)s_1 - ... - (k_m - k_{m-1})s_{m-1} > 0$.*

This Lemma has the following important consequence which is stated as a Lemma.

Lemma 12 *Assume that the last non-trivial derived system of a dynamic feedback linearizable (in the sense of previous Lemma) control system $I^{(k_m+k+1)}$ contains $\pi(\leq p)$ generators and that the structure equations of $I^{(k_m+1)}$ are the first from the bottom of the derived flag to depend on du. Then there exist p integers $\{v_1, ..., v_p\}$ defined by*

$$\begin{array}{ll}
v_i = k + 1 & 1 \leq i \leq s_m \\
v_i = k_m - k_{m-1} + k + 1 & s_m + 1 \leq i \leq s_m + s_{m-1} \\
\vdots & \\
v_i = k_q - k_{q-1} + k + 1 & \sum_{j=q}^{m} s_j + 1 \leq i \leq \sum_{j=q-1}^{m} s_j (= \pi) \\
v_i = k_{q-1} - k_{q-2} + k & \sum_{j=q-1}^{m} s_j + 1 \leq i \leq \sum_{j=q-2}^{m} s_j \\
\vdots & \\
v_i = k_2 - k_1 + k & p - s_1 + 1 \leq i \leq p.
\end{array}$$

The above Lemma indicates that dynamic feedback linearizable systems are associated with a set of generalized controllability indices that must add to the dimension of the original state-space. A generalized *normal form* suitable for the analysis of dynamic feedback linearization has been identified. Next the conditions under which a nonlinear system can be transformed to such a generalized *normal form* are stated.

Theorem 13 *A control-affine nonlinear system is dynamic feedback linearizable by dynamic extensions of state feedback transformations $v = \varphi(x,u)$ if and only if there exists a set of generators adapted to the derived flag associated with a precompensator, P, that belongs to the class described in Lemma 11 such that*
(1) the bottom derived system is trivial
(2) the generators satisfy the congruencies

$$\begin{aligned} d\omega^{q+1}_{v_q-j} &\equiv dt \wedge \omega^{q+1}_{v_q-j+1} \\ &\vdots \qquad\qquad \mod I^{(j+1)}, dv_1, \ldots, dv_q \\ d\omega^{p}_{v_{m_j}-j} &\equiv dt \wedge \omega^{p}_{v_{m_j}-j+1} \end{aligned} \tag{B.5}$$

where $m_j = dim(I^{(j)}/I^{(j+1)})$ is the number of towers with at least $j+1$ rows for $1 \leq j \leq k_m - k_1 + k + 1$, and $q = p - s_1 - \ldots - s_l$ with

$$k_{l-1} - k_1 + 1 \leq j \leq k_m - k_1$$

when $1 \leq l < m$ and

$$k_m - k_1 + 1 \leq j \leq k_m - k_1 + k + 1$$

when $l = m$.

Theorem 13 provides a test for dynamic feedback linearizability for control-affine nonlinear control systems. The application of this theorem to identify dynamic feedback linearizability was illustrated on the 4-state, 2-input example in Chapter 3.